A* Node Search and Nonlinear Optimization for Satellite Relative Motion Path Planning

Ian Edward Peter Connerney

Thesis submitted to the faculty of the
Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Aerospace Engineering

Jonathan T. Black, Chair
Kevin K. Schroeder
Daniel D. Doyle

September 9, 2021
Blacksburg, Virginia

Keywords: Relative Motion, Path Planning, Trajectory Optimization, RPO, A* Search

A* Node Search and Nonlinear Optimization for Satellite Relative Motion Path Planning

Ian Connerney

(ABSTRACT)

The capability to perform rendezvous and proximity operations about space objects is central to the next generation of space situational awareness. The ability to diagnose and respond to spacecraft anomalies is often hampered by the lack of capability to perform inspection or testing on the target vehicle in flight. While some limited ability to perform inspection can be provided by an extensible boom, such as the robotic arms deployed on the space shuttle and space station, a free-flying companion vehicle provides maximum flexibility of movement about the target. Safe and efficient utilization of a companion vehicle requires trajectories capable of minimizing spacecraft resources, e.g., time or fuel, while adhering to complex path and state constraints. This paper develops an efficient solution method capable of handling complex constraints based on a grid search A* algorithm and compares solution results against a state-of-the-art nonlinear optimization method. Trajectories are investigated that include nonlinear constraints, such as complex keep-out-regions and thruster plume impingement, that may be required for inspection of a specific target area in a complex environment. This work is widely applicable and can be expanded to apply to a variety of satellite relative motion trajectory planning problems.

A* Node Search and Nonlinear Optimization for Satellite Relative Motion Path Planning

Ian Connerney

(GENERAL AUDIENCE ABSTRACT)

The ability of one satellite to perform actions near a second space satellite or other space object is important for understanding the space environment and accomplishing space mission goals. The development of a method to plan the path that one satellite takes near a second satellite, such that fuel usage is minimized and other constraints satisfied, is important for accomplishing mission goals. This thesis focuses on developing a fast search and path planning solution method capable of handling complex constraints that can be applied to satellite relative motion operations. The solution method developed in this thesis is then compared to an existing solution method to determine the efficiency and accuracy of the method.

TABLE OF CONTENTS

List of Tables vi
List of Figures vii
1 Introduction 1
1.1 Overview 1
1.2 History of Rendezvous and Proximity Operations 2
1.2.1 First Rendezvous & Docking Missions 2
1.2.2 Push for Autonomy 2
1.2.3 On Orbit Servicing 5
1.3 Current Approach 5
1.4 Challenges 6
1.4.1 Collision Avoidance 6
1.4.2 Thruster Plume Impingement 7
1.5 Research Questions, Task, and Scope 8
1.6 Research Methodology 9
2 Problem Background 10
2.1 Relative Satellite Motion 10
2.1.1 Relative Motion Frames 10
2.1.2 Hill-Clohessy-Wiltshire Equations of Relative Satellite Motion 11
2.1.3 Impulsive Burn Rendezvous using CW Targeting 14
2.1.4 Relative Orbital Elements Parameterization 17
2.2 Optimal Control Theory 18
2.2.1 Optimal Control Problem Overview 18
2.3 Solution Methods Literature Review 20
2.3.1 Optimization Based 21
2.3.2 Solution Approaches 22
2.3.3 Metaheuristic & Random Sampling 23
2.3.4 Challenges 26
3 A* Search 28
3.1 Overview and Description of Algorithm 28

3.1.1 Graph Discretization 28
3.1.2 General Description 28
3.1.3 Properties of the Heuristic Estimate 29
3.1.4 Node Expansion 30
3.1.5 Collision Avoidance 30
3.1.6 Algorithm 30
3.1.7 Simple Path-Length Example 33
3.2 Applications to Relative Motion Problem 35
3.2.1 Grid Generation 35
3.2.2 Heuristic Function 37
3.2.3 Simple Example Utilizing Delta-V Heuristic 41
4 Direct Collocation 43
4.1 Introduction 43
4.1.1 Transcription 44
4.2 Applied to Rendezvous and Proximity Operations 46
4.2.1 Collision Avoidance 47
4.2.2 Thruster Plume Impingement 47
4.2.3 Simple Example 48
5 Selected Scenarios 50
5.1 Simple Waypoint Maneuver 50
5.2 Complex Waypoint Maneuver 53
5.3 Injection into NMT Ellipse 55
5.4 Algorithm Comparison in 100 Randomized Environments 57
5.4.1 Solution Methods 57
5.4.2 Problem to be Solved 58
5.4.3 Results 59
6 Conclusion 66
6.1 Summary 66
6.2 Future Work 67
Bibliography 69

List of Tables

Table Page

Table 1: Autonomous RPO Programs and Missions 3

Table 2: Problem Properties for Simple Waypoint Maneuver 51

Table 3: Solution Results for Simple Waypoint Maneuver 53

Table 4: Problem Properties for Complex Waypoint Maneuver 53

Table 5: Solution Results for Complex Waypoint Maneuver 55

Table 6: Problem Properties for Injection into NMT 55

Table 7: Solution Results for Natural Motion Trajectory Injection 57

Table 8: Problem Properties for 100 Runs 58

List of Figures

Figure Page

Figure 1: RIC reference frame for two satellites in proximity operations, centered at the target satellite. 12

Figure 2: Visual representation of CW Targeting algorithm. 15

Figure 3: Simple two-dimensional grid-based example of A* Search algorithm 32

Figure 4: Solution trajectory for simple 2D path planning problem utilizing a distance based heuristic estimate 34

Figure 5: Visual representation of 2-burn CW Targeting algorithm. Burn magnitude and direction shown in red at each position (filled circle), resulting trajectory shown with a dashed line. 38

Figure 6: Visual representation of Convex Optimization Multi-Burn Rendezvous algorithm. Burn magnitude and direction shown in red at each position (filled circle), resulting trajectory shown with a dashed line. 41

Figure 7: Solution trajectory for simple 2D path planning problem utilizing the Adapted A* Search algorithm described in this section. An admissible heuristic based on the solution to a simple convex optimization sub-problem is used. 42

Figure 8: Solution trajectory for simple 2D path planning problem utilizing solved via the direct trapezoidal collocation method described in this section. 48

Figure 9: Trajectory solutions to the simple waypoint maneuver example problem. (Left) Trajectory generated via the Adapted A* Search algorithm. (Right) Trajectory generated via the Direct Collocation nonlinear optimization method. 52

Figure 10: Trajectory solutions to the waypoint maneuver with complex keep out regions and thruster plume impingement constraints. (Left) Trajectory generated via the Adapted A* Search algorithm. (Right) Trajectory generated via the Direct Collocation nonlinear optimization method. 54

Figure 11: Trajectory solutions to natural motion trajectory example. (Left) Trajectory generated via the Adapted A* Search algorithm. (Right) Trajectory generated via the Direct Collocation nonlinear optimization method. 56

Figure 12: Problem set-up for a complex waypoint maneuver in an example randomized environment. The goal node lies at (0,0,0) and the start node at (-20, -20, -5) in the relative reference frame. 58

Figure 13: Comparison of solution times required of the various methods for 100 random environments using a box and whisker plot. 59

Figure 14: Box and whisker plot for the delta-V values of the various methods for 100 random environments 62

Figure 15: Comparison plot of the delta-V vs computational solution time for the A* Search and Direct Collocation algorithms in each of the 100 random environments 64

Chapter 1

Introduction

1.1 Overview

This introduction is intended to provide a basic understanding of autonomous spacecraft trajectory optimization, demonstrate its need as an enabling technology for future spacecraft missions, and describe the challenges associated with trajectory design that motivate the need for advancement in trajectory planning methods.

Spacecraft trajectory planning refers to the process of generating a dynamically feasible state and control plan for the spacecraft to execute in order to accomplish a goal. In this thesis, the process of *relative* satellite trajectory planning (satellite trajectory planning relative to another spacecraft or object in orbit) will be described in depth along with a comprehensive investigation into relevant solution methods with a focus on *autonomous* relative guidance methods. Relative satellite trajectory planning can be thought of as the trajectory planning portion of a standard rendezvous and proximity operations (RPO) mission. An RPO mission is often described by the motion of one satellite, usually referred to as the "deputy" or "follower", with respect to the second, normally uncooperative, satellite (or other space object) referred to as a "chief" or "target." In this research we will be considering only the final phases of an RPO mission, the "close range rendezvous operations" which by definition begins at a relative distance of a few kilometers to the chief [1].

Here *autonomous* refers to fast and efficient trajectory planning methods that can execute quickly and on-board the satellite, reducing or eliminating the need for communication with the ground. Autonomous satellite rendezvous and proximity operation trajectory design is an enabling technology for the next generation of spacecraft missions. Autonomous RPO trajectory design has

been cited often by both NASA and DARPA as crucial for the future of military, civil, and commercial spacecraft; as in, for example, NASA's 2021 Technical Roadmap for the future [2]–[5].

1.2 History of Rendezvous and Proximity Operations

There exists a rich history of missions utilizing rendezvous and proximity operations to achieve mission goals. Here, a brief timeline and description of the most important RPO missions is presented, including both domestic missions and international space missions, and the more recent commercial endeavors.

1.2.1 First Rendezvous & Docking Missions

The early manned Vostok spacecrafts are potentially the first example of orbital rendezvous and proximity operations. In 1962 Vostok 4 came within 6.5km of Vostok 3 and then shortly thereafter in 1963 Vostok 6 came within 5km of Vostok 5. Not long after this feat, the Gemini program, tasked with demonstrating technologies necessary for lunar landing, led to the remarkably successful close-range rendezvous between Gemini VI and Gemini VII in 1965 during which a relative distance of just 0.3m to 90m between the two spacecraft was maintained for three orbits. Rendezvous and proximity operations continued to be central to both domestic and international space programs, with sustained and extensive use during the Suyoz and Apollo programs [6].

1.2.2 Push for Autonomy

Many of these early missions relied on cooperation between multiple spacecraft and man-in-the-loop relative navigation, an inherently time-consuming and more expensive approach that introduces constraints on the size of the spacecraft and the scope of the mission.

Autonomous rendezvous has the potential to greatly reduce costs associated with rendezvous missions, allowing for a much wider scope of missions free of manual navigation and manned spacecraft.

Russian space probes Phobos 1 and Phobos 2 present an early example of autonomous rendezvous and proximity operations. In 1988 Phobos 1 and 2 were launched with the intent of an unmanned rendezvous with Mars' moon Phobos, maintaining a relative distance of 30-50m from the moon's surface. Phobos 1 met an early demise prior to its planned rendezvous when it was issued an errant command that turned off its attitude control system, resulting in loss of solar power and mission failure [7]. The Phobos 2 mission also failed before it could close on Phobos and deploy a lander due to an onboard computer malfunction. One more recent example, though also rife with failures, is the Hayabusa (2003 – 2010) asteroid sample return mission which utilized autonomous navigation and guidance for station keeping and asteroid sampling. Despite some successes, faulty design logic and long communication times with Earth resulted in the loss of the Hayabusa's mini-lander "MINERVA" intended to survey the surface of the asteroid [8].

Other technology development programs and satellite missions demonstrating autonomous rendezvous and proximity operations are included in the table below.

Table 1: Autonomous RPO Programs and Missions

Mission/Program	Description
DARPA SUMO/FREND (2002) [9]	DARPA funded technology development program to expand critical technology for autonomous rendezvous and capture.
AFRL's XSS-10 (2003) [10]	Low-cost satellite designed to demonstrate optical line of sight relative spacecraft guidance on-orbit for autonomous proximity operations.
AFRL's XSS-11 (2005) [10]	Low-cost satellite developed to test autonomous rendezvous and proximity operations. Focused on testing fully autonomous relative guidance systems.

NASA's DART (2005) [11]	NASA spacecraft intended to test autonomous station keeping, close approach, relative circumnavigation, and collision avoidance. Mission ended early after a low-velocity collision with the target spacecraft.
DARPA Orbital Express (2007) [4]	Autonomous on-orbit servicing and satellite relative repositioning technology demonstration. Tested a servicing satellite prototype on-orbit.
PRISMA (2010)) [12], [13]	Swedish satellite demonstrating formation flying and rendezvous technologies. Performed autonomous formation flying, homing and rendezvous, and precision proximity operations.
ESA TanDEM-X (2010) [14]	Two-satellite constellation flying in 200-500m relative separation to provide Earth terrain and elevation data. Demonstrated autonomous relative station keeping.
DARPA Phoenix (2011) [15]	Program designed to develop on-orbit robotic servicing capabilities. Simulated autonomous rendezvous and grapple in high-fidelity testbed.
DARPA RSGS (2015-Current) [16]	DARPA technology development program to enable satellite inspection and robotic servicing for GEO satellites.
Northrop Grumman's MEV-1 and MEV-2 (2019 and 2021) [17], [18]	On-orbit-servicing spacecraft that successfully refueled the Intelsat spacecraft. Demonstrates commercial applications for rendezvous and docking technology.

Other demonstrative technologies utilizing trajectory planning for RPO include the SPHERES, free-flying satellites aboard the International Space Station [19], [20], and The Aerospace Corporation's Aerocubes [21].

Future missions needing autonomous rendezvous and proximity operation capabilities include NASA's Lunar Gateway [22], (continued) International Space Station, and NASA OSAM-1 (On-Orbit Servicing, Assembly and Manufacturing) [23]. These next generation missions will include on-orbit-refueling, sample return, in-space-inspection, in-space-assembly (ISA), and other close proximity operations.

1.2.3 On Orbit Servicing

Much recent attention has focused on On-Orbit-Servicing (OOS), activities conducted by a space vehicle on orbit that provide some benefit to another spacecraft. Examples of on-orbit-servicing include refueling, orbit modification, routine maintenance, subsystem repair or replacement, vehicle assembly, etc. On-orbit-servicing has many potential benefits including increasing satellite lifetime, mission flexibility, mission scalability (larger missions supported via ISA and OOS), and ultimately reduced cost [10], [24],[25],[26].

Advances in autonomous relative motion trajectory planning are key to increasing capabilities associated with future RPO and OOS missions. Specifically, trajectory planners capable of conforming to complex constraints (e.g., collision avoidance and passive safety) are essential for the future of spacecraft missions. These trajectory planners must be efficient enough to execute autonomously on-board the spacecraft in real time (seconds) and must make efficient use of spacecraft resources (e.g., fuel) in the process.

1.3 Current Approach

Close-range rendezvous trajectories have remained relatively simple over the years. Trajectories typically used for close range rendezvous are straight-line trajectories utilizing continuous thrust to maintain a "straight" approach under the influence of orbital dynamics. These are referred to as 'glideslope' trajectories and typically approach from either the radial or along-track directions. These methods employ an analytically calculated trajectory that prescribes a continuous and decreasing thrust perpendicular to the direction of travel, necessary in order to maintain a straight-line approach [1]. This method uses prohibitively large amounts of fuel (depending on time of flight and distance to the chief spacecraft) and must be individually tailored to each specific maneuver.

Additional approaches to generating trajectories for rendezvous and proximity operations are discussed at length in the literature review in section 2.3. Generally speaking, these trajectory

planners either cannot incorporate complex constraints such as those discussed in section 1.4 or require long computational times, reducing the potential of such planners to be run on-board.

1.4 Challenges

Challenges to current methods for trajectory design include the use of nonlinear objective functions (minimum fuel), collision avoidance, active and passive safety considerations, thruster limitations and thruster plume considerations, and uncertain maneuver duration. Many current solution methods rely on linear constraints and convex problems which are no longer applicable with more complex mission design.

The ability to handle optimization problems with these kinds of constraints is central to advancing rendezvous and proximity operations. This motivates the design of a fast and efficient solution method capable of handling complex constraints.

1.4.1 Collision Avoidance

Collision avoidance is the most important constraint to consider when evaluating trajectory planning algorithms for autonomous rendezvous and proximity operations. An accidental collision is the worst outcome for a spacecraft mission, causing damage or destroying one or both spacecraft involved resulting in mission failure. The ability to accurately and robustly incorporate collision avoidance constraints into a fast and efficient algorithm for trajectory planning is necessary for advancing autonomous proximity operation capabilities. This constraint is generally formulated such that:

$$r_{sc} \notin X_{obs} \tag{1.1}$$

Where r_{sc} denotes the position vector of the spacecraft and X_{obs} denotes a complex obstacle region within the relative frame. Obstacles in this region can be another spacecraft, orbital debris, or other area for the follower spacecraft to avoid, potentially related to sunlight constraints of the target

spacecraft, sensor blind spots, etc. In this way, basic collision avoidance constraints can approximate a series of more complicated constraints.

It is perhaps simplest to express the keep-out-regions in X_{obs} as ellipsoids, relating to uncertainties in the position of the object or region to avoid. If needed, additional complex keep-out-regions can be constructed by combining several simple shapes though this work focuses on ellipsoids for mathematical simplicity.

For many problems, X_{obs} becomes a complex region making collision avoidance constraints non-convex and the related trajectory planning problem inherently difficult to solve. Here, convex relates to a problem type in which there is one global minimum (think the quadratic function). Convex problems are much easier to solve and can be solved much more quickly.

1.4.2 Thruster Plume Impingement

Thruster plume impingement is another complex constraint necessary for many spacecraft proximity operations. Exhaust gasses from thruster firings form a plume that can interact with nearby objects or surfaces, referred to as thruster plume impingement. Plume impingement may cause disturbing forces, unwanted heating, and contamination of target spacecraft surfaces [27], [28]. Contamination or damage to optical equipment or sensors on the target spacecraft is undesirable and necessitates a solution method capable of handling thruster plume impingement constraints. These typically take the form of restricting the thruster capability of the follower spacecraft when near the target spacecraft.

Many different models of the thruster plume exist. For this work a simple model approximating the thruster plume as a line segment with length proportional to the magnitude of the thrust is used. This thrust length, with some additional tolerance, is then checked within the collision avoidance algorithm for impact.

Simplification of the thruster plume model is used in this work to help reduce algorithm computational time. Incorporating an extremely accurate thruster plume model and/or exhaustively

checking for collision is computationally inefficient and prohibitive for a fast and efficient algorithm capable of being autonomously run on-board. That being said, if required, both of the solution methods developed and described in this work are capable of incorporating an additionally complex thruster plume model for plume impingement.

Even a basic implementation of a thruster plume impingement constraint model is non-convex and requires a solver capable of incorporating non-convex constraints in order to generate a plume impingement free trajectory.

1.5 Research Questions, Task, and Scope

- Research questions:
    - Can randomness-based path planners be used to generate optimal RPO trajectories subject to complex non-convex constraints?
    - How does the Adapted A* Search algorithm presented in this literature compare to other state of the art solution methods for solving nonlinear optimization problems?
    - Can the Adapted A* Search algorithm be applied to consistently produce better initial guess trajectories for use with nonlinear optimizers?
- Research Task:
    - Adapt the A* Search algorithm for minimum fuel trajectory optimization in the presence of complex nonlinear constraints.
    - Use Direct Collocation methods to solve complex relative motion trajectory optimization problems in the presence of nonlinear constraints.
    - Compare results from the Adapted A* Search algorithm developed in this thesis to the Direct Collocation solution method.
- Research Scope:
    - This research addresses rendezvous and proximity operations conducted by one follower satellite about a second target satellite. This research considers highly constrained trajectory planning problems including multiple, complex, keep out regions and thrust plume impingement. Though additional complex and nonlinear

constraints exist for the relative motion trajectory planning problem, this work focuses only on complex keep out regions and thruster plume impingement.

1.6 Research Methodology

Two solution methods for optimal trajectory planning will be presented in this work with adaptations to the relative motion trajectory planning problem and applied to several in-space-inspection maneuvers:

1. Adapted A* Search Algorithm: path planning algorithm that utilizes a heuristic in order to efficiently search a graph of nodes for the lowest cost path from the start node to the goal node.

2. Direct Collocation: Nonlinear trajectory optimization method in which state and control variables are approximated using piecewise continuous polynomials between a set number of nodes used as inputs to a nonlinear programming solver.

Chapter 2

Problem Background

2.1 Relative Satellite Motion

This section introduces the relative satellite motion equations used within this paper. Various definitions and parameterizations of relative satellite motion will be investigated to develop a baseline understanding for the general relative motion problem.

2.1.1 Relative Motion Frames

Cartesian

One common coordinate frame for describing the position of one satellite relative to another target or chief satellite is known as the 'Radial, In-Track, Cross-Track' (RIC) reference frame which is often centered at the target satellite. Following the conventions in Vallado's text, the basis vectors for this frame are defined as following. The Radial direction or X axis points outwards from the target satellite along the radius vector from the Earth's center to the satellite as it moves through the orbit. The Cross-Track or Z axis points along the angular momentum vector normal to the orbital plane of the target satellite found via the cross product between position and velocity. The In-Track or Y axis completes the set of basis vectors and is defined perpendicular to both the radial and normal such that the set is a right-handed coordinate system (the Y axis will be in the same direction as the target's velocity vector though not necessarily aligned for elliptical orbits). Additionally, there is no requirement that the center of the frame be at an actual satellite; our target (or center of the relative reference frame) can be at any point of interest following some orbital trajectory [29].

Curvilinear

A relative motion frame based on curvilinear spherical coordinates has been shown to be more accurate than the cartesian representation due to unmodeled nonlinear terms from higher order dynamics. These nonlinear effects can be reduced if a curvilinear system is used, particularly in the in-track and cross-track directions [30]–[32]. In practice, the use of a curvilinear reference frame produces identical equations of motion to the standard HCW equations with the linear distances (x, y, z) replaced by arc-length displacements $(\delta r, a_0\theta_r, a_0\phi_r)$. A full derivation and comparative analysis of the HCW equations in curvilinear coordinates can be found in references [30], [31].

2.1.2 Hill-Clohessy-Wiltshire Equations of Relative Satellite Motion

The HCW (Hill-Clohessy-Wiltshire) equations describe the motion of a 'follower' or 'deputy' satellite relative to a 'target' or 'chief' satellite in the target satellite's RIC reference frame, centered on the target spacecraft. These equations are a result of a linearization of the circular-restricted nonlinear equations of relative motion which are valid for relative distances (between the target and follower spacecraft) much smaller than the distance between the target and the center of the central body (Earth) and for smaller timescales.

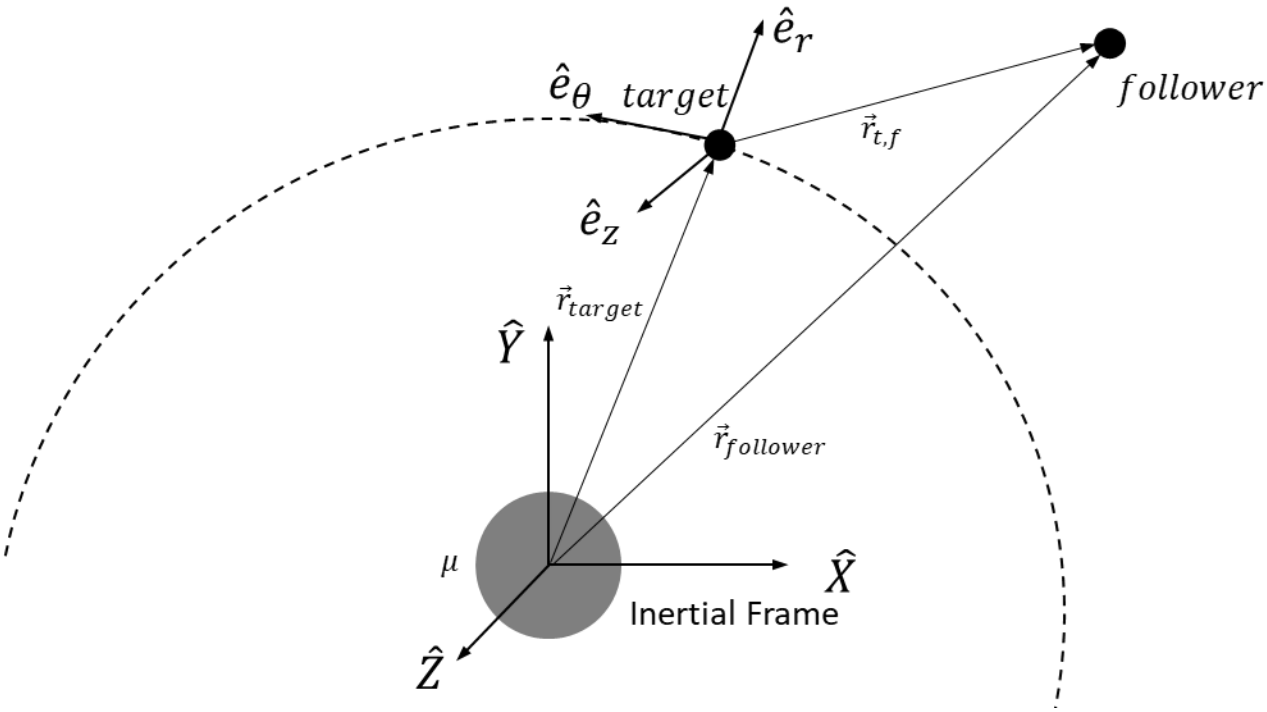

Figure 1: RIC reference frame for two satellites in proximity operations, centered at the target satellite.

The target satellite is restricted to be in a circular orbit and other parameters for the target's orbit are not required to represent the relative motion. The unforced equations of relative motion in the curvilinear RIC frame are as follows:

$$\begin{aligned} \delta\ddot{r} - 2a_0 n\dot{\theta}_r - 3n^2\delta r = 0 \\ a_0\ddot{\theta}_r + 2n\delta\dot{r} = 0 \\ \ddot{\phi}_r + n^2\phi_r = 0 \end{aligned} \tag{2.1}$$

Where $(\delta r, \theta_r, \phi_r)$ are the relative coordinates along the radial, curvilinear in-track, and curvilinear cross-track directions respectively and a_0 corresponds to the semi-major axis of the target satellite. For simplicity we set (x, y, z) equal to their respective arc length displacements $(\delta r, a_0\theta_r, a_0\phi_r)$:

$$\begin{aligned} x = \delta r \\ y = a_0\theta_r \end{aligned} \tag{2.2}$$

$$z = a_0 \phi_r$$

And add control accelerations $a_x, a_y,$ and a_z such that the equations of motion become:

$$\begin{aligned} \ddot{x} - 2n\dot{y} - 3n^2 x &= a_x \\ \ddot{y} + 2n\dot{x} &= a_y \\ \ddot{z} + n^2 z &= a_z \\ where\ n &= \sqrt{\frac{\mu}{a^3}} \end{aligned} \tag{2.3}$$

Where $\hat{x}$ points in the radial direction, $\hat{y}$ points along the orbit of the target satellite (in-track), and $\hat{z}$ points in the curvilinear out-of-plane or cross-track direction. n is the mean-motion of the target satellite's orbit and is defined as the square root of the ratio of the central body's standard gravitational parameter, $\mu = Gm$, where G is the gravitational constant, and a is the semi-major axis of the target. $a_x, a_y,$ and a_z are the components of the acceleration of the follower due to external forces (i.e., thrusting) in the radial, in-track, and cross track directions respectively. Note that these equations are the same as the initial HCW equations with respect to a cartesian reference frame. The differences being that the in-track, x, and cross-track, z, are now measured along arc-lengths, which reduces the effects of neglected nonlinearity in determining the initial conditions and from the expansion of the differential gravity in the derivation [30].

Solutions to the homogenous HCW equations (for which accelerations $a_x, a_y,$ and a_z are 0, i.e., no thrust) were first derived by Clohessy and Wiltshire in [33] and can also be found in various texts such as [29] and [34]. These solutions can be expressed with the following state transition matrix (STM), Φ, which is a function of only the transfer time, t.

$$\begin{bmatrix} \delta r(t) \\ \delta v(t) \end{bmatrix} = \Phi(\mathrm{t}) \cdot \begin{bmatrix} \delta r(t_0) \\ \delta v(t_0) \end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix}\delta r(t)\\ \delta v(t)\end{bmatrix} = \begin{pmatrix} x(t)\\ y(t)\\ z(t)\\ \dot{x}(t)\\ \dot{y}(t)\\ \dot{z}(t)\end{pmatrix} \tag{2.5}$$

$$\Phi(\mathrm{t}) = \begin{bmatrix} 4-3\cos(nt) & 0 & 0 & \frac{\sin(nt)}{n} & \frac{2\,(1-\cos(\mathrm{nt}))}{n} & 0\\ 6(\sin(nt)-nt) & 1 & 0 & \frac{2(\cos(nt)-1)}{n} & \frac{4\sin(nt)-3nt}{n} & 0\\ 0 & 0 & \cos(nt) & 0 & 0 & \frac{\sin(nt)}{n}\\ 3n\sin(nt) & 0 & 0 & \cos(nt) & 2\sin(nt) & 0\\ 6n(\cos(nt)-1) & 0 & 0 & -2\sin(nt) & 4\cos(nt)-3 & 0\\ 0 & 0 & -n\sin(nt) & 0 & 0 & \cos(nt)\end{bmatrix} \tag{2.6}$$

Where t is the transfer time – the change in time between the initial state and the final state.

2.1.3 Impulsive Burn Rendezvous using CW Targeting

In the paper by Clohessy and Wiltshire [33] a method for rendezvous using impulsive burns is described, which can also be shown in terms of the HCW STM. This method can be used to calculate the two impulsive burns (one at initial state and one at the final state) to rendezvous with another space object in the relative motion frame. These impulsive burns represent the only two impulsive burns (one at initial state and one at final state) to rendezvous with the final state and are therefore the optimal burns for a specified transfer time. This rendezvous approach is visualized in Figure 2 shown below.

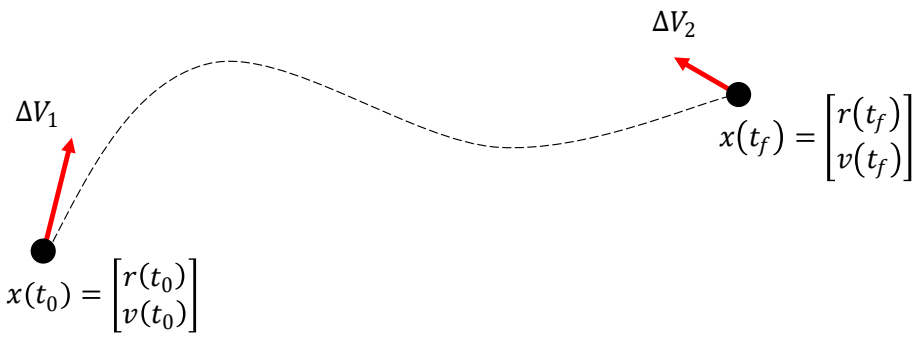

Figure 2: Visual representation of CW Targeting algorithm.

The magnitudes and directions of the two burns can be easily derived from the HCW state transition matrix shown in section 2.1.2 on the HCW equations of motion. First for simplicity and readability we divide the HCW STM into four sub-matrices representing the quadrants of the initial STM matrix.

$$\Phi = \begin{bmatrix} 4 - 3\cos(nt) & 0 & 0 & \frac{\sin(nt)}{n} & \frac{2\,(1 - \cos(\mathrm{nt}))}{n} & 0 \\ 6(\sin(nt) - nt) & 1 & 0 & \frac{2(\cos(nt) - 1)}{n} & \frac{4\sin(nt) - 3nt}{n} & 0 \\ 0 & 0 & \cos(nt) & 0 & 0 & \frac{\sin(nt)}{n} \\ 3n\sin(nt) & 0 & 0 & \cos(nt) & 2\sin(nt) & 0 \\ 6n(\cos(nt) - 1) & 0 & 0 & -2\sin(nt) & 4\cos(nt) - 3 & 0 \\ 0 & 0 & -n\sin(nt) & 0 & 0 & \cos(nt) \end{bmatrix} \quad (2.7)$$

$$\Phi = \begin{bmatrix} \Phi_{rr} & \Phi_{rv} \\ \Phi_{vr} & \Phi_{vv} \end{bmatrix} \quad (2.8)$$

These sub-matrices are then:

$$\Phi_{rr} = \begin{bmatrix} 4 - 3\cos(nt) & 0 & 0 \\ 6(\sin(nt) - nt) & 1 & 0 \\ 0 & 0 & \cos(nt) \end{bmatrix} \qquad \Phi_{rv} = \begin{bmatrix} \frac{\sin(nt)}{n} & \frac{2\,(1 - \cos(\mathrm{nt}))}{n} & 0 \\ \frac{2(\cos(nt) - 1)}{n} & \frac{4\sin(nt) - 3nt}{n} & 0 \\ 0 & 0 & \frac{\sin(nt)}{n} \end{bmatrix}$$

$$\Phi_{vr} = \begin{bmatrix} 3n\sin(nt) & 0 & 0 \\ 6n(\cos(nt) - 1) & 0 & 0 \\ 0 & 0 & -n\sin(nt) \end{bmatrix} \qquad \Phi_{vv} = \begin{bmatrix} \cos(nt) & 2\sin(nt) & 0 \\ -2\sin(nt) & 4\cos(nt) - 3 & 0 \\ 0 & 0 & \cos(nt) \end{bmatrix} \tag{2.9}$$

The position and velocity at a transfer time *t* can be represented in terms of the initial conditions and the sub-matrices defined above:

$$r(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \Phi_{rr} r(t = 0) + \Phi_{rv} v(t = 0) \tag{2.10}$$

$$v(t) = \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{pmatrix} = \Phi_{vr} r(t = 0) + \Phi_{vv} v(t = 0) \tag{2.11}$$

Using this notation and by rearranging terms, we can calculate impulsive two burn rendezvous trajectories between any two points described in the relative RIC frame for a fixed transfer time, t. The two impulsive burns ΔV_1 and ΔV_2 occurring at $t = 0$ and t respectively are then defined by the following:

$$\Delta V_1 = {\Phi_{rv}}^{-1} \left(r(t) - \Phi_{rr}\big(r(t = 0)\big) \right) - v_0^- = v_o^+ - v_0^- \tag{2.12}$$

Where v_o^+ is the required velocity at time $t = 0$ required to travel from the initial position $r(t = 0)$ to the rendezvous location at the final time $r(t)$. v_0^- is the current velocity at time $t = 0$. The difference between the two then produce the required delta-V burn to embark on the rendezvous trajectory.

The second impulsive burn can then be calculated by propagating the state forward and taking the difference between the propagated state and the required delta-V to rendezvous.

$$\Delta V_2 = v_f^+ - (\Phi_{vr} r(t=0) + \Phi_{vv} v_o^+) \tag{2.13}$$

Where v_f^+ is the desired velocity at time t for rendezvous.

Thus, the total ΔV requirement for the impulsive burn maneuver becomes:

$$\Delta V_{total} = |\Delta V_1| + |\Delta V_2| \tag{2.14}$$

2.1.4 Relative Orbital Elements Parameterization

The solutions to the HCW equations as shown above can be re-parameterized as a set of relative orbital elements that fully characterize the follower's motion as shown in [35]. This set of relative orbital elements provides a clearer geometric insight into the relative motion of the follower and how the unforced motion evolves with time. These relative orbital elements are expressed in terms of instantaneous cartesian state elements (as opposed to initial conditions) in the LVLH frame as follows:

$$x_r = 4x + \frac{2\dot{y}}{n} \tag{2.15}$$

$$y_r = y - \frac{2\dot{x}}{n} \tag{2.16}$$

$$a_r = \sqrt{\left(6x + \frac{4\dot{y}}{n}\right)^2 + \left(\frac{2\dot{x}}{n}\right)^2} \tag{2.17}$$

$$Er = atan2\left(\frac{2\dot{x}}{n}, 6x + \frac{4\dot{y}}{n}\right) \tag{2.18}$$

$$A_z = \sqrt{z^2 + \left(\frac{\dot{z}}{n}\right)^2} \tag{2.19}$$

$$\psi = atan2\left(z, \frac{\dot{z}}{n}\right) \tag{2.20}$$

Where x_r and y_r describes the radial and in-track coordinates of the instantaneous center of motion, a_r describes the semi-major axis of the relative ellipse, Er describes the relative eccentric anomaly, and A_z and ψ describe the amplitude and phase angle of the cross-track harmonic motion.

For unforced motion, $x_r, a_r,$ and A_z remain constant while y_r varies linearly with time as a function of the secular drift rate negatively proportional to the radial coordinate x_r. The angular relative orbital elements Er and ψ change at a constant angular rate equal to the target satellite's mean motion, n.

If $x_r = 0$ then there is no secular drift of the relative motion in the y (along-track) direction which is a convenient property to exploit for low-cost long-term bounded motion about the target satellite. This constraint can be shown in terms of initial conditions by setting $x_r = 0$ in the equations above giving:

$$\dot{y} = -2nx \tag{2.21}$$

This results in a projected 2D ellipse in the orbital plane with length $2a_r$ in the in-track direction and width a_r in the radial direction.

2.2 Optimal Control Theory

2.2.1 Optimal Control Problem Overview

Trajectory planning problems for satellite RPO involve describing a desired path for the satellite in terms of state (position, velocity) and control (thrust) variables in order to meet some objective as a function of time. In a similar manner, some problems may require attitude control as well, which would expand the state variables to include some form of attitude representation (Euler angles, quaternions, modified Rodriguez parameters), their rates of change, and control torques.

In order to find the "best", or optimal, trajectory we need to select inputs to the system (our control variables) that minimize a chosen performance index, which may be total control thrust, time of

flight, any other describable quantity, or some combination thereof. Additionally, we may require that our trajectory or control be subject to a series of constraints.

We employ a set of trajectory optimization (or optimal control) methods in order to solve this problem. Following surveys [36]–[38], and a Direct Collocation tutorial [39], an optimal control problem is generally formulated as the following:

First and arguably the most important constraint on our optimization problem is the system dynamics, which may be linear (HCW equations) or non-linear and describes how our system changes with time. Given a set of n differential equations:

$$\dot{x}(t) = f(x(t), u(t), t) \tag{2.22}$$

Where $x(t) \in \mathbb{R}^n$ is the continuous time vector of state variables, $u(t) \in \mathbb{R}^m$ is the continuous time vector of control variables, and t is the independent time variable, $t \in [t_0, t_f]$.
The optimal control problem is then to determine the state $x(t)$ and control $u(t)$ that minimizes a chosen performance index, generally of the form:

$$\min_{t_0, t_f, x(t), u(t)} J\left(t_0, t_f, x(t_0), x(t_f)\right) + \int_{t_0}^{t_f} \omega(x(\tau), u(\tau), \tau) d\tau \tag{2.23}$$

The optimization problem is also subject to a series of constraints. The path constraint enforces restrictions along the trajectory. This type of constraint may be used to ensure that the trajectory does not intersect a keep-out region. These will be expressed as:

$$g\big(t, x(t), u(t)\big) \leq 0 \tag{2.24}$$

Another common type of constraint is a non-linear boundary constraint, which restricts the initial and final state of the system. This can be used to mandate an exact relation between final state variables as is described further in the section on zero drift trajectories and relative orbital elements.

$$h\left(t_0, t_f, x(t_0), x\left(t_f\right)\right) \leq 0 \tag{2.25}$$

We also specify constant limits on the state or control which represent absolute limits throughout the entire trajectory. These are expressed on the trajectory as:

$$\begin{aligned} x_{low} \leq x(t) \leq x_{upp} \\ u_{low} \leq u(t) \leq u_{upp} \end{aligned} \tag{2.26}$$

And on the initial and final state and time as:

$$\begin{aligned} t_{low} \leq t_0 < t_f \leq t_{upp} \\ x_{0,low} \leq x(t_0) \leq x_{0,upp} \\ x_{f,low} \leq x(t_f) \leq x_{f,upp} \end{aligned} \tag{2.27}$$

Most trajectory optimization problems are too complex to be solved analytically, so we will focus on numerical methods for solving optimal control and/or trajectory planning problems.

2.3 Solution Methods Literature Review

In recent years there has been a large amount of research effort in the field of optimal control towards spacecraft relative motion planning and trajectory optimization, well summarized in surveys by Betts, Conway, GuoQiang, and Rao, [36]–[38], [40] and with a survey by Topputo focusing on direct transcription methods for low-thrust space trajectories [41]. Additional surveys by Goerzen, L. Yang, and Y. Yang focus on random sampling and graph search methods to solve the optimal motion planning problem. This approach has been used in the robotics field to provide extremely fast solutions to high degree-of-freedom problems and has also been applied to guidance for unmanned aerial vehicles (UAVs), autonomous vehicles (2007 DARPA Urban Challenge) [42] and spacecraft trajectory planning [43]–[45]. In the sections below, the most promising solution methods will be summarized and presented along with the relevant literature.

2.3.1 Optimization Based

Convex Optimization

Simple spacecraft relative motion path planning problems can be expressed as convex optimization problems and solved via simple gradient descent methods. Benefits of convex optimization include quicker, guaranteed, convergence to the global minimum without the need for an initial guess. Convex optimization is not suitable, however, for problems with complex non-linear and non-convex constraints such as those posed by complex keep out regions. Some recent efforts using Mixed Integer Linear Programming (MILP) [46], [47] and clever approximations of non-convex keep-out-ellipses as a combination of rotating hyperplanes [48], [49] allow for solving slightly more complex relative motion path planning problems. Mixed Integer Linear Programming is a solution method that splits the trajectory optimization problem into several convex regions which allows for the use of binary constraints to represent keep-out-regions.

Direct Collocation

Perhaps the most promising group of methods for solving complex satellite relative motion path planning problems are Direct Collocation methods. These methods discretize the continuous trajectory into state and control variables at discrete timesteps with space between nodes being represented with polynomial splines. Constraints, including dynamical constraints, are then posed on each of the nodes and the resulting nonlinear program (NLP) can be solved via a nonlinear programming solver. The process of converting the continuous time problem into its' discretized version is known as transcription. Many different kinds of transcription methods exist determining how and where nodes are chosen as well as how the state and control is approximated between nodes. Note that this approach is more costly than convex optimization, but it allows for nonlinear and non-convex constraints that better describe the satellite relative motion path planning problem. Drawbacks of using Direct Collocation methods are that the resulting nonlinear program (NLP) both requires and is sensitive to the quality of the initial guess used with the solver. Poor initial guesses may not converge to feasible solutions even if one exists and the nonlinear solver may also get stuck in local minima. For this reason, one of the main difficulties in implementing a Direct Collocation method is in constructing the initial guess trajectory. [36]–[39], [50] suggest

first solving a more simple but related problem analytically or by exploiting evolutionary algorithms that are better suited to finding global minima.

[51] Uses a type of Direct Collocation known as pseudospectral or orthogonal collocation that involves the use of Legendre polynomials as global basis functions. Initial guesses for the NLP are generated by solving approximate problems involving parameter optimization of finite sequences of burns and coasts via a particle swarm optimization problem and a genetic algorithm.

2.3.2 Solution Approaches

Direct Methods

Direct methods transcribe the continuous optimal control problem into a parameter optimization problem composed of state and control variables at discrete timesteps. Depending on assumptions made and the resulting problem type, this parameter optimization problem can then be solved via a linear or non-linear programming solver (NLP). Constraints (including dynamics constraints) are applied at each discrete timestep (or node) in the form of either equality or inequality constraints. The conversion process between continuous time optimal control problem and the discretized version described above is also known as transcription [38], [39].

There are many different types of transcription methods which affect how and where nodes are chosen as well as how a function (state or control) is approximated between nodes.

Indirect Methods

Indirect methods represent another set of solution methods for solving optimal control problems. Indirect methods approach the optimal control problem by first analytically constructing the necessary and sufficient conditions for optimality and then solving the problem. These conditions are typically discretized, and the problem is then solved via a numerical solver to find an optimal solution. This approach is occasionally referred to as “optimize then discretize” as opposed to the direct method’s approach of “discretize then optimize.” The main benefit of indirect methods when

compared to direct methods is that the indirect methods are generally more accurate with a more reliable error estimate though they are often slower and much more difficult to pose. For these reasons this work will focus mainly on direct method solutions and randomness-based solutions [37], [39].

2.3.3 Metaheuristic & Random Sampling

Particle Swarm Optimization & Genetic Algorithms

Evolutionary algorithms have relatively recently been applied to spacecraft trajectory optimization. Evolutionary algorithms are numerical optimizers that use methods similar to those found in nature to find an optimal set of parameters to characterize the solution to an optimal control problem. Some examples include evolutionary algorithms such as GA (MATLAB genetic algorithm), PSO (Particle swarm), and ant colony optimization. These solution methods can be used in contrast to gradient based optimization methods (such as MATLAB's FMINCON) for problems in which an initial guess may be difficult to generate or where chances of falling into a local minimum is likely. These algorithms solve these problems by incorporating randomness into the solution method, but they may be slower or require more iterations to converge. Evolutionary algorithms have the drawback that the solution must be relatively simple and capable of being described by a relatively small set of parameters which may not fully represent the original problem complexity.

One benefit to these methods is that they can be used to provide an initial guess for more complex solution methods such as Direct Collocation which rely on a good initial guess to converge quickly. Evolutionary algorithms are better suited to finding the global minimum and when used as an initial guess they may help more complex methods converge to a global minimum instead of a local minimum.

Pontani and Conway have extensively explored the application of particle swarm optimization to space trajectories in [50], [52]–[54]. Additionally, evolutionary algorithms have been used to provide an initial guess for the more complex Direct Collocation methods in [51], [55], [56].

Random Sampling and Graph Search Path Planning

Sampling and graph search path planning algorithms have been long used for motion planning in other fields and have more recently been applied to spacecraft relative motion trajectory planning. These methods utilize random (or deterministic) sampling in order to generate paths through the solution space. Properties of these algorithms allow for very fast runtime when compared to trajectory optimization methods such as Direct Collocation and optimality guarantees. These algorithms are extremely efficient at solving path planning problems in high dimensional spaces with keep out regions. A wide range of sampling-based algorithms are described in the literature with various adaptions to different kinds of problems. Some common random-sampling and node-search based algorithms will be described in the following section along with applications to the satellite relative motion trajectory planning problem. Random-sampling methods (such as rapidly exploring random tree or RRT) continually generate samples in the state space and continuously attempt connections between nearby samples. RRT* (the * denotes asymptotic optimality) provides an improvement on RRT in that if a lower cost path becomes available through a newly sampled node, that path is added to the tree and the 'old' higher cost path is removed.

In [57] an RRT* algorithm was used to generate an initial path in a complex spacecraft relative motion planning problem. This initial path was then refined using a basic local optimizer.

Fast Marching Tree or FMT* is another sampling-based motion planning algorithm which was applied to the spacecraft relative motion problem in [58]. This algorithm boasts the ability for shifting a large portion of computation off-line to better support on-board applications.

Probabilistic Road Map (PRM) takes one batch of samples and generates an interconnected graph between the start and goal nodes. Graph search algorithms such as A* and Dijkstra's algorithm are then used to search the graph for the shortest (or lowest cost) path. A* utilizes a heuristic, typically distance to the goal node, and a cost, current path length, in order to bias the search towards the goal by prioritizing the expansion of nodes that can reach the goal with lower cost. If the heuristic is well formulated (estimated cost to reach goal is never higher than actual cost to reach goal) then A* ensures optimality.

A* based methods have been used in the past satellite relative motion path planning problems starting with the basic rendezvous problem in [59]–[61] which similarly use a cost function composed of a fuel estimate in order to find low delta-V solutions, though obstacles are not considered. Reference [62] Uses a similar A* based motion planning algorithm to solve the basic rendezvous problem with plume constraints in the presence of obstacles where new nodes are generated by randomly sampling feasible controls based on a normal distribution. A local gradient descent algorithm is run on the solution to further optimize the path. This proposed method provides a foundation for an A* based solution methodology but has some shortcomings. By using randomized expansion and an inadmissible heuristic, completeness and optimality cannot be guaranteed and the solver is forced to evaluate far more nodes than necessary in order to generate a path. Reference [63] makes some improvements on the algorithm in [62], expanding it to use the Tshauner-Hempel equations of motion and provides different weight functions for tree expansion. Reference [64] uses an A* search algorithm for spacecraft proximity planning maneuvering between waypoints around a 3D sphere in conjunction with PD controller for smoothing the initial trajectory. The cost function used is based entirely on the angular separation between nodes and likely does not do a good job at minimizing delta V. Reference [65] also attempts to solve the autonomous spacecraft rendezvous and docking problem by utilizing an A* search method. This paper expands on previous work to include different heuristics (minimum time and minimum path length) though makes no mention of minimum fuel trajectories. An additional collision detection method is implemented based on preemptively eliminating nodes which only generate infeasible child nodes to improve performance.

Another random sampling-based method described in [66] utilizes a spherical expansion-based sampling algorithm to explore the workspace. Once a path is found between the start and goal node a locally optimal trajectory is then found using sequential convex programming. It should be noted here that the initial spherical expansion method does not take into account system dynamics, and the sequential convex programming step minimizes path length along the trajectory and not fuel – which is a less useful objective for spacecraft relative motion problems as fuel or time constraints are typically prioritized over path length. Additionally, the use of convex programming prohibits additional complex constraints such as thruster plume impingement constraints.

The main benefit to these types of methods is the solution speed. While direct/indirect methods may better approach a true optimal solution with enough time, metaheuristic randomness-based methods are often used to more quickly find an adequate (though not necessarily optimal) solution in a much quicker timeframe. However, many of these algorithms have asymptotic optimality in that as the number of samples taken approaches infinity the lowest cost path approaches the true optimal – though in practice this usually requires large computational time. Another benefit to these randomness-based sampling algorithms is the ability to avoid getting stuck in local minima (which is a shortcoming of potential field and gradient based optimization methods).

One disadvantage of using these methods is that they generally produce jagged, non-optimal, paths unless they are run for a very long amount of time. One way to improve on the jaggedness of these paths is to run a local optimizer or smoothing function on the output of the random-based path planning method using gradient descent or other method.

2.3.4 Challenges

Challenges to current methods for trajectory design include the use of nonlinear objective functions (e.g., minimum fuel), collision avoidance, active & passive safety considerations, thruster limitations & thruster plume considerations, and uncertain maneuver duration. Many current solution methods rely on linear constraints and convex solution methods which are no longer applicable to more complex mission design.

The ability to handle optimization problems with these kinds of complex constraints is central to advancing rendezvous and proximity operations and enabling autonomous in-space-inspection. This motivates the design of a fast and efficient solution method capable of handling complex constraints.

1. Nonlinear Objective Functions & Dynamics
2. Collision Avoidance

3. Active & Passive Safety
    a. Safety Ellipses
    b. Approach Corridor
4. Thruster limitations & Thrust Plume Impingement
5. Uncertain Maneuver Duration

Chapter 3

A* Search

3.1 Overview and Description of Algorithm

The A* Search is a path planning algorithm that utilizes a heuristic in order to efficiently search a graph of nodes for the lowest path from the start node to the goal node.

3.1.1 Graph Discretization

The A* Search algorithm is a grid-based path planning algorithm and functions by searching over a graph of states. The graph is traditionally generated by discretizing the state space into finite 'cells' of some resolution where each cell represents all of the state spaces that fall within it. The graph is constructed by building a vertex for each cell and by adding edges between vertices that can be feasibly connected via admissible actions. This process transforms the optimal path planning problem into a graph search problem over the discretized state space. The constructed graph can often be quite large and in practice is generally created on-demand such that vertices and edges are generated only when the graph search algorithm requires in order to save on storage space and limit the number of nodes evaluated. Optimality guarantees are generally only ensured up to the grid resolution and if the heuristic satisfies additional constraints to be described later.

3.1.2 General Description

The heuristic is a very important piece of the A* search algorithm and the main differentiator between A* and random-sampling based methods (such as RRT* and FMT*). At any moment during the runtime of the algorithm, the exact cost to reach the goal node from any given node is

unknown because the optimal path to reach the goal node has not yet been discovered. The heuristic can then be thought of as an estimate of the cost to reach the goal node (in place of the true cost to reach the goal node) and is typically denoted as $H(n)$. The heuristic will be used to help guide the search. Each node also contains a value for its generation cost, $G(n)$, - the known cost to reach that node through all nodes already in the path. These costs can then be added together to obtain an estimate, $F(n)$, of the total cost to reach the goal along a path through the current node.

$$F(n) = G(n) + H(n) \tag{3.1}$$

By first expanding the search to nodes with a lower total heuristic value the search is directed to seek out paths with the lowest potential overall cost, i.e., 'optimal' paths. Nodes with high cost estimates are expanded last or not at all. Once a path is found to the goal node, that path is known to be the optimal path through our use of the heuristic estimate (all paths with a lower heuristic have already been evaluated and there is no need to evaluate paths with a higher estimate as these provide sub-optimal solutions). This greatly reduces the number of nodes that the search method needs to visit in order to find the optimal path, which has the additional benefit of reducing computational time.

3.1.3 Properties of the Heuristic Estimate

There are some properties of the heuristic cost function estimate that require mentioning. Firstly, the exact cost to reach the goal node is unknown because the optimal path to reach the goal node is also unknown. The heuristic cost estimate must provide a conservative estimate in order to be labelled *admissible*, that is never overestimate the true cost to reach the goal node in order to ensure optimality. Admissible heuristics also need to expand less nodes in order to find a solution [44]. Additionally, it has been shown that the more *aggressive* (how closely the heuristic estimate approximates the true cost) the heuristic estimate, the better the convergence rate.

In practice the ability of the A* search algorithm to exploit this heuristic estimate allows for the rapid generation of optimal paths faster than other path planning methods.

3.1.4 Node Expansion

Traditionally, at each iteration nodes are expanded by evaluating the cost of neighboring nodes and choosing the lowest cost node to expand to. For motion planning problems which feature kinodynamic constraints it is advantageous to use a different method for node expansion [67]. Because adjacent states do not necessarily reflect reachable states when kinodynamic constraints are added, motion primitives are introduced to better reflect accessibility for systems with constraints on kinematics or dynamics. The use of motion primitives will be discussed at greater lengths in the following section on the A* Search algorithm adapted to the relative motion problem.

3.1.5 Collision Avoidance

An important part of path planning algorithms is the ability to easily incorporate complex collision avoidance constraints to allow for the efficient generation of collision free paths between points. This is accomplished via checking each node against X_{obs}, the set of states in the relative motion frame that obstacles for collision. If the node is checked and determined to be within X_{obs} it is excluded from being added to the heap, effectively removing the node from being expanded to in the future. Accuracy can be improved by interpolating along the trajectory between the nodes to ensure that the entire path is collision free as opposed to just the nodal points at burn locations. The number of nodes to be checked along the path would be determined by specific collision avoidance accuracy requirements and may not be necessary for all applications depending on the distances between nodes. For simplicity and to avoid additional costly collision avoidance checks, the collision avoidance requirement will be relaxed to check only at each of the nodal points within the A* algorithm.

3.1.6 Algorithm

The A* Search algorithm starts as the start node is *expanded*. When expanding a node, the potential successors (i.e., child nodes) are identified through a node expansion technique. Each potential successor is then checked to ensure it does not violate any constraints. If the successor node does not violate any constraints, then its known generation cost, $G(n)$, and heuristic estimate, $H(n)$, are evaluated to obtain the total cost estimate, $F(n)$. The successor node is then added to a data

structure known as a minimum heap which efficiently sorts the nodes into levels based on the value of the total cost estimate, $F(n)$. After all valid successor nodes are added to the heap, the lowest cost node is pulled off the heap and expanded with the process repeating until the goal node is reached.

Figure 3 shows the A* search algorithm in a simple two-dimensional grid discretization in which the heuristic, $H(n)$, is based on the distance to the goal node. At each iteration, the node expanded is shown highlighted in yellow with nodes that have already been visited and expanded shown as black circles. For generation and heuristic cost evaluation, each grid square is 10 units in length and width and the diagonals are approximated as 14 units. At each node, all adjacent nodes (including those along the diagonals) are available for expansion.

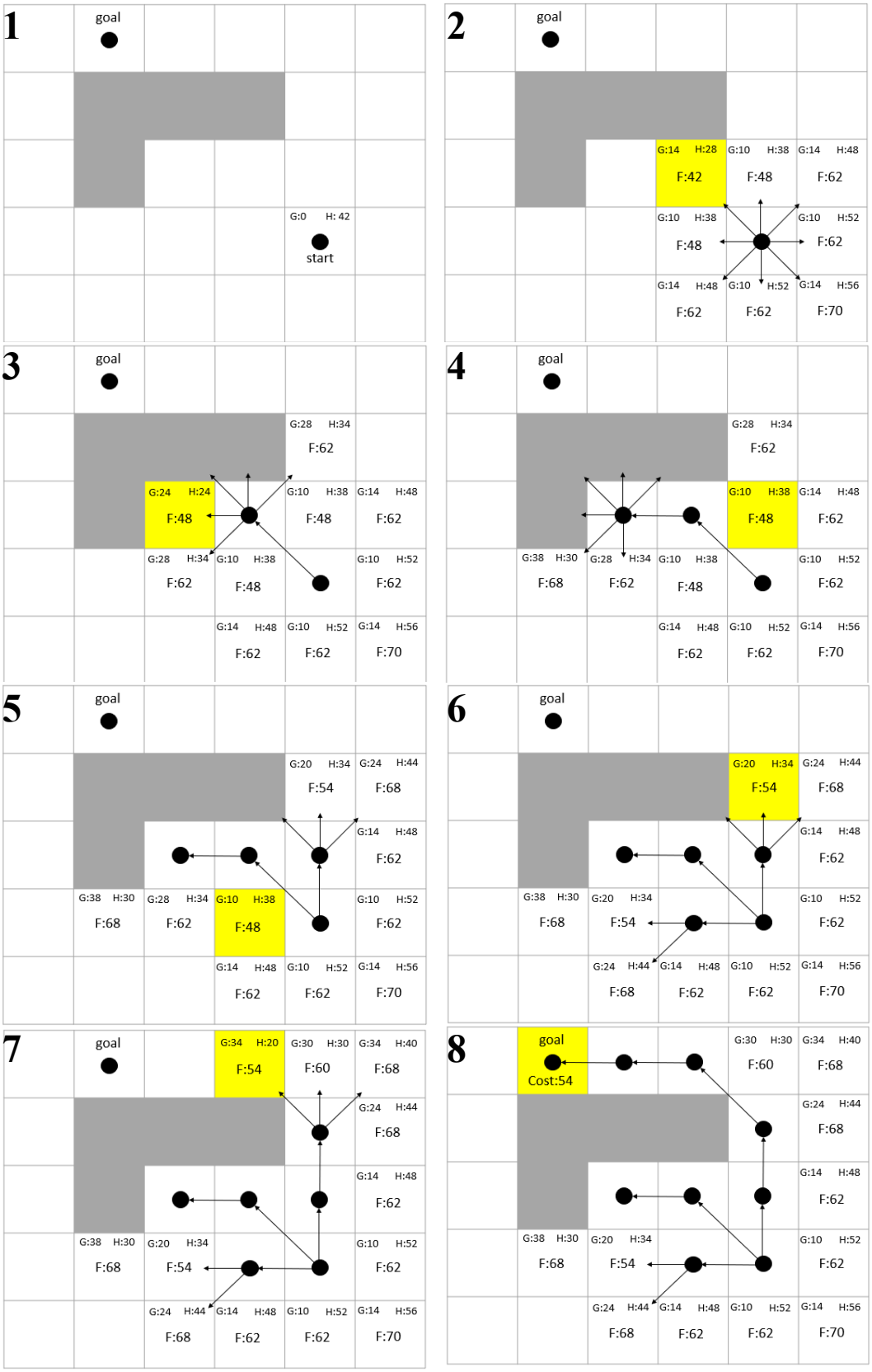

Figure 3: Simple two-dimensional grid-based example of A* Search algorithm

3.1.7 Simple Path-Length Example

To better understand the need for an A* algorithm tailored towards rendezvous and proximity operations, a simple path-length based example as applied to a RPO problem will be described with an analysis of the resulting solution path. This example is based on a simple grid discretization in which (for simplicity) nodes are located every two meters in an XY grid. Neighboring nodes open for expansion are then the eight nodes immediately surrounding the current node with a maximum distance traveled of $2\sqrt{2}$ meters corresponding to the diagonal case. For this simple example our state is composed only of X and Y positions (no velocities or times) though a similar example could be generated in which the state is composed of discretized positions, velocities, and times.

The heuristic used for this simple example is the traditional Euclidian distance heuristic often used for planners minimizing the overall path length.

$$H(n) = \sqrt{\left(x_{goal} - x_n\right)^2 + \left(y_{goal} - y_n\right)^2} \tag{3.2}$$

Where x_{goal} and y_{goal} are the X and Y locations of the goal node and x_n and y_n are the X and Y locations of the current node.

The generation cost for any node is then determined by adding the cost to go from the parent node to the current node to the parent's generation cost.

$$G(n) = G(parent) + G(n, parent) \tag{3.3}$$

Here $G(n)$ is the generation cost of the current node, n, $G(parent)$ is the generation cost of the parent node and $G(n, parent)$ is the actual cost to travel from the parent node to the current node.

A simple collision avoidance constraint is implemented in which the planner considers only nodes that are collision free in the set X_{free}. For this example, there exists one circular keep out region in the center of the coordinate system $[0,0]$ with radius 2.5m.

The problem is set up such that the start location is at $[0,-10]$ and the goal location is at $[0,10]$. For consistency with plots in the relative motion frame and to better understand the motion in the plots, the X axis (plotted vertically) is taken from the X direction in the relative motion frame which corresponds to the radial direction. Negative X values correspond to a follower that is closer to the Earth than the chief (i.e., "below" the chief). The Y axis (plotted horizontally) then corresponds to the curvilinear In-Track direction, with positive values taken in the direction of the chief's velocity vector.

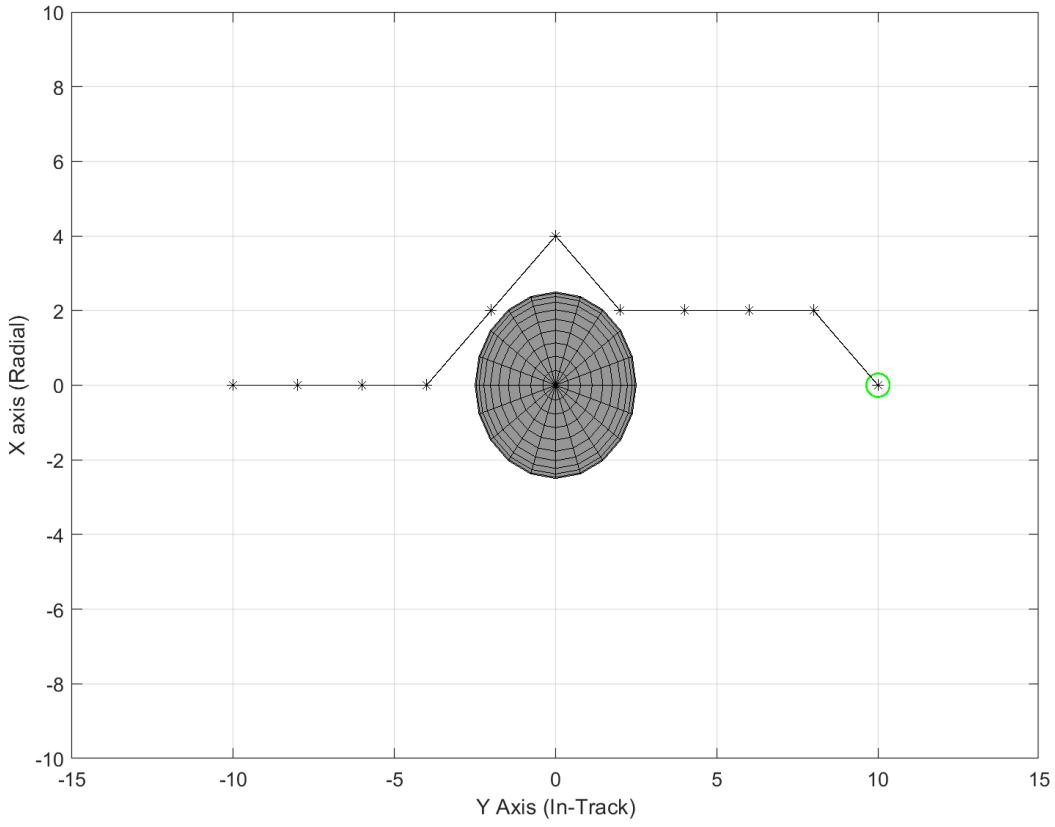

Figure 4: Solution trajectory for simple 2D path planning problem utilizing a distance based heuristic estimate

Clearly the path generated via standard path planning techniques utilizing a distance-based heuristic is sub-optimal when applied to a kinematically constrained object, in our case a spacecraft, that is attempting to navigate along the trajectory. This can be seen in the jagged paths that such a planner produces resulting in larger delta-V required to remain on the path. Additionally, this solution contains no information as to how a spacecraft would actually attempt to fly the trajectory in terms of velocities or accelerations along the trajectory, which is necessary for ensuring that bounds on velocities, accelerations, time constraints etc. are satisfied. It can be seen now that though this method produces viable collision free paths, a different solution method is required incorporating both the dynamics and kinematics of the problem as well as additional, more complex, constraints (i.e., plume impingement).

3.2 Applications to Relative Motion Problem

The application of the A* Search path planning algorithm to rendezvous and proximity operations in the relative motion frame presents two major challenges. The first and most basic lies with representing the state space of the kinodynamic problem (i.e., a motion planning problem subject to kinematic and dynamic constraints). The second major challenge is the development of a heuristic to represent the cost-to-go to the goal from any node. This section will describe the specific implementation of the Adapted A* Search algorithm to the relative motion path planning problem.

3.2.1 Grid Generation

There exist two main approaches to grid generation for grid-search based problems, conventional grid and dynamic grid generation.

Conventional Grid Generation

Conventional grid discretization creates a uniform grid of finite cells wherein the entire state space (composed of valid positions, velocities, and times) is discretized. Node expansion is then done by evaluating neighboring nodes in the grid. This approach is impractical for the relative motion

problem as it would require a very large number of nodes in order to accurately represent every possibility within the state space – many of which would represent sub-optimal or even physically unreachable states within our kinodynamic system. The extremely large number of nodes needed to represent the entire state space of the system would also drastically slow down our solver leading to computational times that are unrealistic for on-board applications. Additionally, because relative motion trajectories are naturally curved, many velocity changes (i.e., small thrusts) would be needed to pass through points in the discretization space resulting in the generation of even larger, non-optimal, delta-V values.

Dynamic Grid Generation

The second approach is a more dynamic form of grid generation and node expansion based upon motion primitives and natural relative motion orbital mechanics. Nodes for expansion from the current node are chosen by applying a predefined set of impulse burns to the current node and propagating the state through a fixed timestep with the HCW state transition matrix described in the earlier section on relative satellite motion. This method ensures that each node is dynamically feasible and represents a reachable state under bounds on impulsive thrust magnitude. This leads to far fewer nodes evaluated and much quicker computational times.

One consideration when using this method is how to determine which and how many impulsive thrusts to consider for propagating and generating new nodes. For some problems with minimal keep out regions or other constraints it may make the most sense to use only impulsive burns that are close to the optimal burn calculated with the heuristic. However, there may exist a scenario in which the problem is heavily constrained in terms of keep out regions or other constraints in which staying near the calculated optimal trajectory is infeasible and generates no paths that avoid the keep out regions even though one may exist. For heavily constrained scenarios it may be advantageous to use a wide variety of impulsive burn amplitudes and directions. In practice the number of impulsive burns to use is often decided by the computational time constraints of the algorithm and the directions and magnitudes of these burns is decided by the constraints on the specific problem.

There are two important burns to consider propagating in every case; these would be the no thrust condition corresponding to a coast phase and the burn corresponding to the minimum cost trajectory returned by the heuristic function (if available). The no burn case is important because it implicitly allows for trajectories to be composed of burn-coast-burn sequences which are more likely to approach an optimal solution under Pontryagins Maximum Principle [68]. The optimal burn case corresponding to the minimum cost trajectory returned by the heuristic function is important for convergence – once the follower is on the optimal trajectory it will remain on this trajectory until it reaches the goal node or violates a constraint. Depending on the choice of heuristic function, determining the optimal burn may not be possible. This will be explored in the following discussion of applicable heuristic functions.

3.2.2 Heuristic Function

The standard straight-line Euclidian distance heuristic function is usually a bad representation of the true cost to fly a trajectory in space – which relates more to the fuel expended and is closely tied to a change in velocity (e.g., delta-V). In order to remedy this and develop a solution method to generate minimum fuel trajectories we focus on heuristic functions capable of estimating the delta-V needed to reach the goal node. In the next section we will discuss two solution methods for generating a delta-V based heuristic. The first is a 'naïve' delta-V estimate based on the CW Targeting algorithm derived from the HCW state transition matrix. A second, novel, method for delta-V estimation is developed in this section based on solving a simple convex optimization sub problem for an N-burn rendezvous.

CW Targeting Heuristic

The CW Targeting heuristic is a longstanding method for determining the optimal two burn rendezvous using the HCW equations of motion and state transition matrix. This is the optimal and only two burn rendezvous for a specific transfer time. A better measure of the overall two burn optimal rendezvous can be determined by searching over possible transfer times.

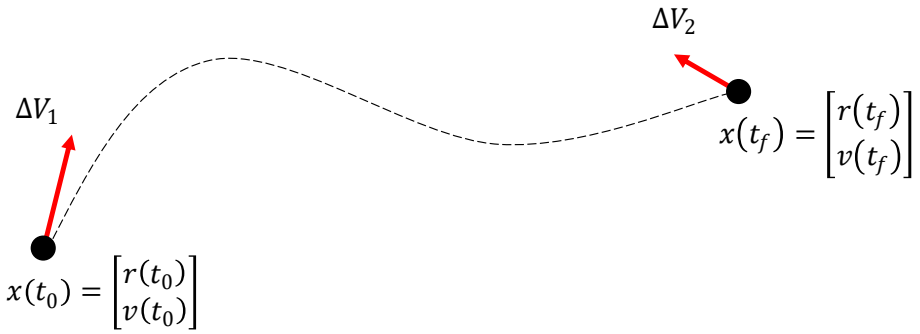

Figure 5: Visual representation of 2-burn CW Targeting algorithm. Burn magnitude and direction shown in red at each position (filled circle), resulting trajectory shown with a dashed line.

The two delta-V burns and the total delta-V for the CW Targeting two impulse rendezvous maneuver as described in section 2.1.3 is given again below.

$$\begin{aligned} \Delta V_1 &= \Phi_{rv}{}^{-1}\left(r(t) - \Phi_{rr}\left(r(t=0)\right)\right) - v_0^- = v_o^+ - v_0^- \\ \Delta V_2 &= v_f^+ - (\Phi_{vr} r(t=0) + \Phi_{vv} v_o^+) \\ \Delta V_{total} &= \Delta V_1 + \Delta V_2 \end{aligned} \tag{3.4}$$

This heuristic has been used in past satellite motion planning problems such as those described in the literature review section in order to connect two randomly sampled points. One important note about this heuristic is that it is not admissible due to the fact that the two-burn rendezvous is not necessarily the lowest delta-V rendezvous for a fixed rendezvous time. A lower cost rendezvous can be completed with up to four different burns for the planar problem or six different burns for the full three-dimensional problem. Work by Neustadt states that for linear dynamics with an *n* dimensional state space, at most *n* impulsive burns are needed to produce an optimal transfer [69]. A more detailed explanation of various direct and indirect methods to solve for the *n* impulsive burns are given in [70]–[75]. These sources were used as a general reference for the development of the convex optimization heuristic described in the next section.

Convex Optimization Sub Problem Heuristic

The motivation behind the convex optimization sub problem heuristic is to design a delta-V heuristic that is admissible for use with the Adapted A* Search algorithm. Admissibility of the heuristic ensures optimality to the level of discretization, known as resolution optimality, and reduces the number of nodes that need to be checked [45]. In order for the heuristic to be admissible it needs to never overestimate the actual cost to reach the goal node. One way to do this is similar to the direct method approach in [39] in which the times and delta-V values of each burn are formulated as inputs to a nonlinear optimization problem with additional nonlinear constraints to ensure that the state (relative position, propagated using the HCW linearized dynamics model) matches at each of the burn locations. This problem can be solved via a nonlinear optimizer such as MATLAB's FMINCON to give a true optimal minimum delta-V for n number of impulsive burns at any burn time and for any final transfer time. This nonlinear optimization solution represents a true lower bound on the delta-V needed but is too complex and inefficient to be used as a heuristic within the Adapted A* Search algorithm.

The Adapted A* Search algorithm allows burns to occur only at a discretized and finite amount of fixed timesteps. By also constraining the final time of rendezvous we can cast the discretized optimal multi-burn rendezvous as a convex second order cone problem (SOCP) instead of a nonlinear optimization problem. This approach was shown in [46] to closely approximate the true optimal solution if enough timesteps are used. As a heuristic for the Adapted A* Search algorithm, only timesteps where burns can occur need be considered in the convex optimization problem to provide an admissible and aggressive heuristic. These timesteps can be easily identified by looking at the setup of the Adapted A* Search algorithm. Due to the convex nature of the problem, numerical solvers can more easily find a global optimal solution in a very short amount of time, making this heuristic potentially feasible for on-board applications.

This convex optimization problem is formulated as follows:

$$\min_{\Delta V_j, \sigma_j} J = \sum_{j=1}^{N} \sigma_j$$

$$\begin{aligned} &Subject\ to: \\ &\Delta V_{x,j}^2 + \Delta V_{y,j}^2 + \Delta V_{z,j}^2 \leq \sigma_j^2 \qquad \text{for } j = 1, \dots, N \\ &x_{goal} = \Phi(\mathrm{N}\Delta\mathrm{t}) x_{start} + \sum_{j=1}^{N} \Phi((\mathrm{N} - \mathrm{j})\Delta\mathrm{t}) \begin{pmatrix} \mathbf{0} \\ \Delta V_j \end{pmatrix} \\ &\sigma_{low} \leq \sigma_j \leq \sigma_{upp} \end{aligned} \tag{3.5}$$

Here we include an additional set of slack variables, σ_j, corresponding to the magnitudes of each of the N burns. These variables allow us to express the originally nonlinear objective function as linear, a necessary condition for formulation as a SOCP. In this way a relaxed convex optimization problem is created whose solution is identical to the original problem with a nonlinear 2-norm minimum delta-V objective function [17], [47]. This approach allows us to minimize the total delta-V expenditure where each burn is represented by a single thruster. The only other variables in the solver are then the impulsive burns themselves where:

$$V_j = \begin{bmatrix} \Delta V_{x,j} \\ \Delta V_{y,j} \\ \Delta V_{z,j} \end{bmatrix} \tag{3.6}$$

The timesteps between burns, $\Delta\mathrm{t}$, start state, x_{start}, goal state, x_{goal}, are known a priori. We can then calculate a minimum delta-V for an N-burn impulsive rendezvous between a start state and goal state. This delta-V value is the same as our objective function:

$$\Delta V_{total} = \sum_{j=1}^{N} \sigma_j \tag{3.7}$$

This value can then be used as an admissible heuristic within our A* path planning algorithm, improving convergence, and guaranteeing optimality to the level of discretization. The algorithm is additionally resolution complete in that if a solution exists then the algorithm is guaranteed to find the solution in a finite amount of time. These are very important properties for our autonomous relative motion planning problem and justify the use of a more computationally expensive heuristic estimate. Additional information and proofs for resolution optimality and completeness of the A* algorithm can be found in [45].

A visualization of the convex optimization sub problem can be seen in Figure 6.

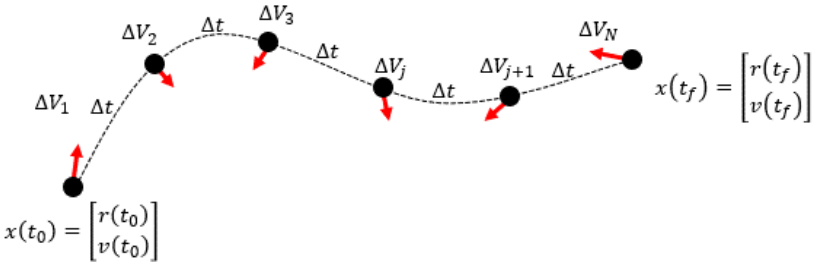

Figure 6: Visual representation of Convex Optimization Multi-Burn Rendezvous algorithm. Burn magnitude and direction shown in red at each position (filled circle), resulting trajectory shown with a dashed line.

3.2.3 Simple Example Utilizing Delta-V Heuristic

In order to better explain the Adapted A* algorithm, a solution to the simple two-dimensional example from the previous section is presented. Here, consistent with the solution method described in this section, the state at each node is composed of position, velocity, and time. Additionally, each node contains information on its generation cost, $G(n)$, corresponding to the actual delta-V requirement to reach that node and a heuristic estimate $H(n)$, corresponding to the solution of the convex optimization sub-problem starting at the current node and rendezvousing with the goal node at the final transfer time. The dynamic grid generation is based on applying a finite series of impulsive burns (including a no-burn coast and the optimal burn calculated with the convex optimization sub problem) to the current node and propagating the state forward with

the HCW STM as described earlier in this section. For this example, a total of 10 discrete burns were considered at each node.

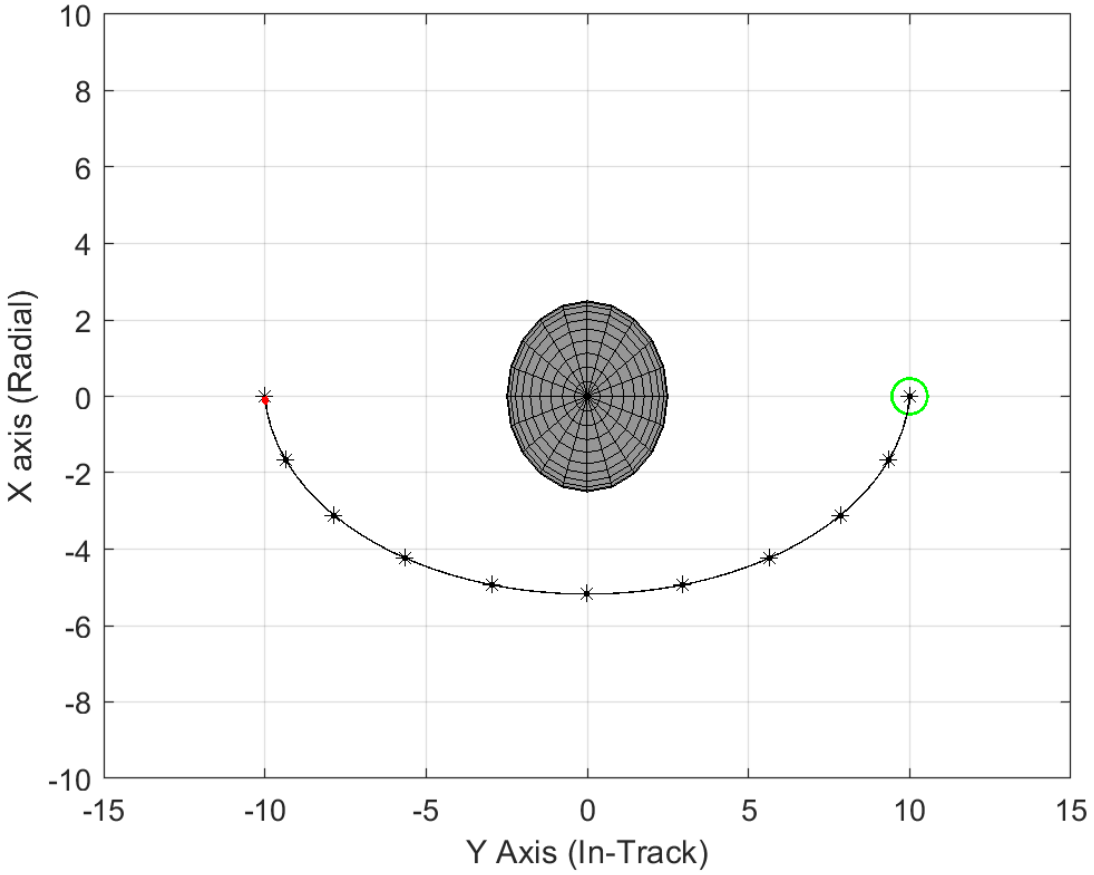

Figure 7: Solution trajectory for simple 2D path planning problem utilizing the Adapted A* Search algorithm described in this section. An admissible heuristic based on the solution to a simple convex optimization sub-problem is used.

As can be seen in the solution trajectory for the simple example in Figure 7, this solution method presents a much better solution in terms of smoothness, corresponding to satisfying kinematic constraints along the path. This solution method also allows for minimizing fuel (delta-V) for a given transfer and provides far more information in terms of how the trajectory is actually flown. The solution method provides a full time history of impulsive burns and coasts between the start and goal node with position, velocity, and time recorded along the way.

Chapter 4

Direct Collocation

4.1 Introduction

The trajectory optimization method applied in this work is known as Direct Collocation in which state and control variables are approximated using piecewise continuous polynomials between a set number of nodes. Properties of these piecewise polynomials allow for the approximation of the trajectory using algebraic equations for integration and incorporating dynamics between discretized nodes or collocation points (known as quadrature). This greatly reduces the computational time required to generate a solution but can introduce some error if higher order or nonlinear dynamics are being approximated by linear or quadratic polynomials between nodes. Additionally, constraints are only required to be satisfied at the collocation points or nodes and not in between. Accuracy can be increased by increasing the number of nodal points along the trajectory or by increasing the order of the polynomial used to approximate the trajectory between nodal points though with a tradeoff in increased computational time. In [76] Herman and Conway demonstrated that higher degree quadrature methods with fewer subintervals showed quicker computational times for the same minimum relative error. In this work, however, a simple trapezoidal method for approximating the state and control variables between collocation points is used. The motivation behind using the trapezoidal method is simplicity and to allow for the maximum number of collocation points throughout the trajectory for a given computational time. Increasing the number of collocation points is important for ensuring adherence to complex constraints such as collision avoidance and thruster plume impingement. Having more collocation points allows the algorithm to better satisfy constraints across the entire trajectory without the need for interpolating between points. More information on the specifics of the constraints considered within this solution method is presented at the end of this section.

4.1.1 Transcription

In order to solve the optimal control trajectory optimization problem via a Direct Collocation method as described above the continuous-time problem is converted into a discretized version that can be solved via a nonlinear programming solver (such as MATLAB's FMINCON function [77]), this process is known as transcription. Following the procedure in Dr. Kelly's Tutorial [39], the trajectory itself is first discretized into a finite set of decision variables that represent the state, position $x(t)$ and velocity $v(t)$, and control, $u(t)$, at specific points in time. This set of decision variables are known as collocation points or nodes.

$$\begin{aligned} t &\rightarrow t_0, \dots, t_k, \dots, t_N \\ x(t) &\rightarrow x_0, \dots, x_k, \dots, x_N \\ v(t) &\rightarrow v_0, \dots, v_k, \dots, v_N \\ u(t) &\rightarrow u_0, \dots, u_k, \dots, u_N \end{aligned} \tag{4.1}$$

Next, the continuous system dynamics are converted into a set of constraints that can be applied to the state and control variables at each of the collocation points. The trajectory (state and control) between nodes is approximated using polynomial splines which allows for computationally efficient integral calculations between nodes which is particularly important for enforcing dynamics constraints and objective function calculation. This is where the quadrature method comes into play. Here trapezoidal quadrature (also known as the trapezoid rule) is used for simplicity making our collocation method trapezoidal collocation.

In trapezoidal collocation the continuous control trajectory and dynamics are approximated as a series of linear splines between nodes. The state trajectory can then be represented by quadratic splines between nodal points (this is due to the integration of the linearly approximated dynamics).

This gives the following equations for the dynamical constraints on the state, x:

$$\begin{gathered} \dot{x} = f \\ \int_{t_k}^{t_{k+1}} \dot{x}\, dt = \int_{t_k}^{t_{k+1}} f\, dt \\ x_{k+1} - x_k \approx \tfrac{1}{2}(t_{k+1} - t_k) \cdot (f_{k+1} + f_k) \end{gathered} \tag{4.2}$$

and for calculating the integral term in our objective function:

$$\int_{t_0}^{t_f} \omega(x(\tau), u(\tau), \tau) d\tau \approx \sum_{k=0}^{N-1} \tfrac{1}{2}(t_{k+1} - t_k) \cdot (\omega_{k+1} + \omega_k) \tag{4.3}$$

Note here that the state, x_k, and control, u_k, at each grid point are decision variables in the non-linear program and $f_k = f(x_k, u_k, t_k)$ is the system dynamics at each collocation point, k.

In addition to the dynamical constraints, there may also be limits on state and control variables, path constraints, and boundary constraints. These constraints are generally handled by enforcing them at specific collocation points. The limits on state and control variables are approximated as:

$$\begin{gathered} x_{low} \leq x_k \leq x_{upp} \\ u_{low} \leq u_k \leq u_{upp} \end{gathered} \tag{4.4}$$

Path constraints are approximated at each node as:

$$g(t_k, x_k, u_k) < 0 \tag{4.5}$$

Boundary constraints are similarly applied at the start and final nodes:

$$h\big(t_0, t_f, x_0, x_f\big) \leq 0 \tag{4.6}$$

The continuous time optimal control problem has now been discretized into a parameter optimization problem and can be solved via a nonlinear programming solver, of which there are many. MATLAB's FMINCON function is used in this work.

4.2 Applied to Rendezvous and Proximity Operations

The Direct Collocation method as previously described can be easily applied to the trajectory planning problem for rendezvous and proximity operations. For consistency the linear HCW dynamics model used with the Adapted A* solution method discussed in the last section is also applied here as a dynamical constraint at each of the collocation points as discussed previously.

The state and system dynamics (consistent with the HCW equations) are then given as:

$$x = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \qquad and\ \dot{x} = f = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ 3n^2x + 2n\dot{y} + a_x \\ -2n\dot{x} + a_y \\ -n^2z + a_z \end{bmatrix} \tag{4.7}$$

The objective function used in this work is a minimum delta-V formulation corresponding to minimum fuel expenditure where the continuous delta-V is integrated via the trapezoidal rule and given as:

$$\min_{t_0, t_f, x_k, u_k} \sum_{k=0}^{N-1} \frac{1}{2}(t_{k+1} + t_k) \cdot (|u_k| + |u_{k+1}|) \tag{4.8}$$

Different objective functions may be used if desired, but the formulation above was considered for consistency with the Adapted A* Search algorithm developed herein.

Collision avoidance and thrust plume impingement constraints are also implemented at each of the collocation points through the path constraint formulation in equation (4.5). Constraints considered match those from the Adapted A* Search algorithm and will be described in the following sections.

4.2.1 Collision Avoidance

For the Direct Collocation solution method collision avoidance will be enforced by checking each of the nodal points against the known locations of obstacles in the state space. For simplicity, within this work each obstacle will be modeled as a three-dimensional ellipsoid as described in the general section on collision avoidance. A nonlinear inequality constraint is then constructed for each three-dimensional ellipsoid and for each collocation point to check for distance from the ellipsoid boundary. This results in three nonlinear inequality constraints per collocation point for each keep out region. Despite the large number of nonlinear constraints, the nonlinear solver is still able to solve the solution in fairly low computational time, further discussed in section 5. This equation closely resembles that of an ellipsoid.

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}+\frac{(z-z_0)^2}{c^2}\geq 1 \tag{4.9}$$

Here x_0, y_0, z_0 represent the location of the center of the ellipsoid and a, b, c represent the half-lengths of the ellipsoid along each axis. The coordinates x, y, z refer to the position of the collocation point being checked for collision. Additionally, the evaluation of this nonlinear constraint (as opposed to a Boolean value for collision detection) allows for the nonlinear program solver to use numerical gradients in solving the constrained collision avoidance problem which improves computational time.

4.2.2 Thruster Plume Impingement

Our simplified thruster plume model can be incorporated into the Direct Collocation algorithm in much the same way as the collision avoidance constraints. At each collocation point the thrust vector (scaled by some length) is checked against the ellipsoid keep out regions using a similar formulation to equation (4.9). Due to the continuous thrusting approximation used within Direct

Collocation it must be noted that this constraint is an imperfect measure of thruster impingement between nodes as the continuous thrusting between nodes is not checked for impingement. There is a trade-off here between including more nodal points for plume impingement detection accuracy and increased computational burden. If desired, additional points between collocation points can be checked using the approximations of the state and control variables consistent with the particular collocation method. In this work, only a simplified thruster plume impingement model was implemented in which only the collocation points are considered for thruster plume impingement.

4.2.3 Simple Example

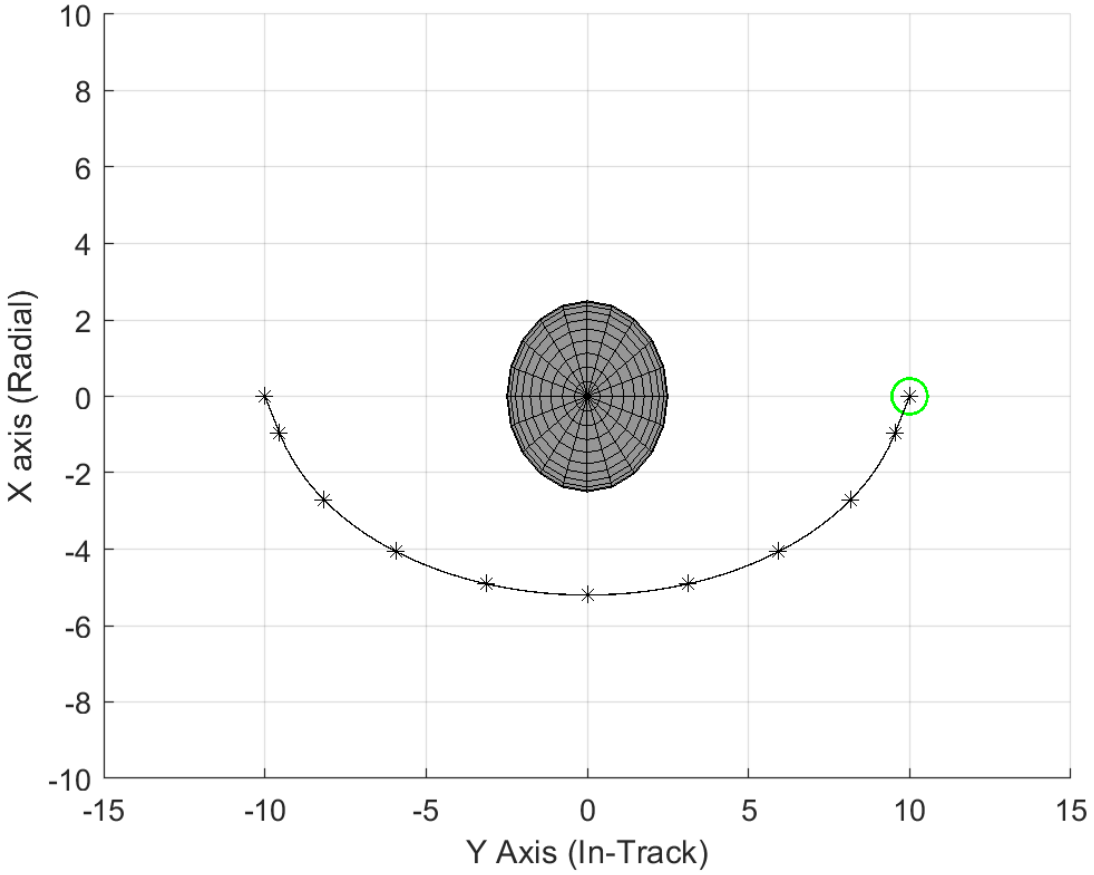

Figure 8: Solution trajectory for simple 2D path planning problem utilizing solved via the direct trapezoidal collocation method described in this section.

This resulting trajectory solution generated using the direct trapezoidal method and a nonlinear optimization solver matches quite closely with the trajectory generated with the Adapted A* Search algorithm described in previous sections. The main differentiator between this trajectory and the trajectory generated with the path planning algorithm is that this trajectory features continuous thrusting as opposed to the impulsive burns in the Adapted A* Search method. This continuous thrusting is responsible for the shorter spacing between the first two and last two nodes as the follower spacecraft is accelerating continuously.

Chapter 5

Selected Scenarios

Simulations for several test cases were performed in order to compare the two algorithms in terms of computational time and solution quality (in terms of delta-V). All simulations were run in MATLAB 2021a on a single computer with Intel(R) Core(TM) i7-6600U CPU processor at 2.60 GHz and with 16 GB of RAM.

Adapted A* Search Set-Up

For the following scenarios, 18 discrete burns at each node were considered for dynamic grid generation. Burns attempted at each node consisted of all combinations of maximum thrust burns in either 1 or 2 combined directions (radial, in-track, or cross-track) and a coast (no-burn). The maximum thrust burn allowed in the cross-track direction was half that for the radial and in-track directions due to the state bounds of the specific problem set up. For the Adapted A* Search using the convex optimization sub problem as a heuristic, the optimal burn was also attempted. This results in a total of 20 burns attempted at each node for the Adapted A* Search and 19 for the CW Targeting A* Search.

5.1 Simple Waypoint Maneuver

The first example problem deals with a simple waypoint maneuver proceeding from 'behind' the target in the in-track direction to 'in-front' of the target in the in-track direction with an ellipsoid keep out region centered at $(0,0,0)$ in the radial, in-track, and cross-track directions respectively. Initially the follower is at $(0,-20,0)$ and is tasked with reaching the goal node at $(0,20,0)$ within a fixed amount of time while minimizing fuel usage (delta-V). Additional problem properties are listed in Table 2. Max thrusts for the impulsive Adapted A* Search and Direct Collocation solution

methods were developed such that the maximum thrust impulsive thrust corresponds to the integral of the maximum continuous thrusting over the same interval to provide some level of delta-V parity between solution methods.

Table 2: Problem Properties for Simple Waypoint Maneuver

Parameter	Chief Orbit (km)	Transfer Time (s)	Follower Mass (kg)	Max Thrust (impulsive) (m/s)	Max Thrust (continuous) (m/s^2)
Value	6791	600	1	.12	.004

Trajectories were generated using the two solution methods described above results can be seen in Figure 9. The Adapted A* Search algorithm solution is plotted on the left while the Direct Collocation nonlinear optimization method solution is shown on the right.

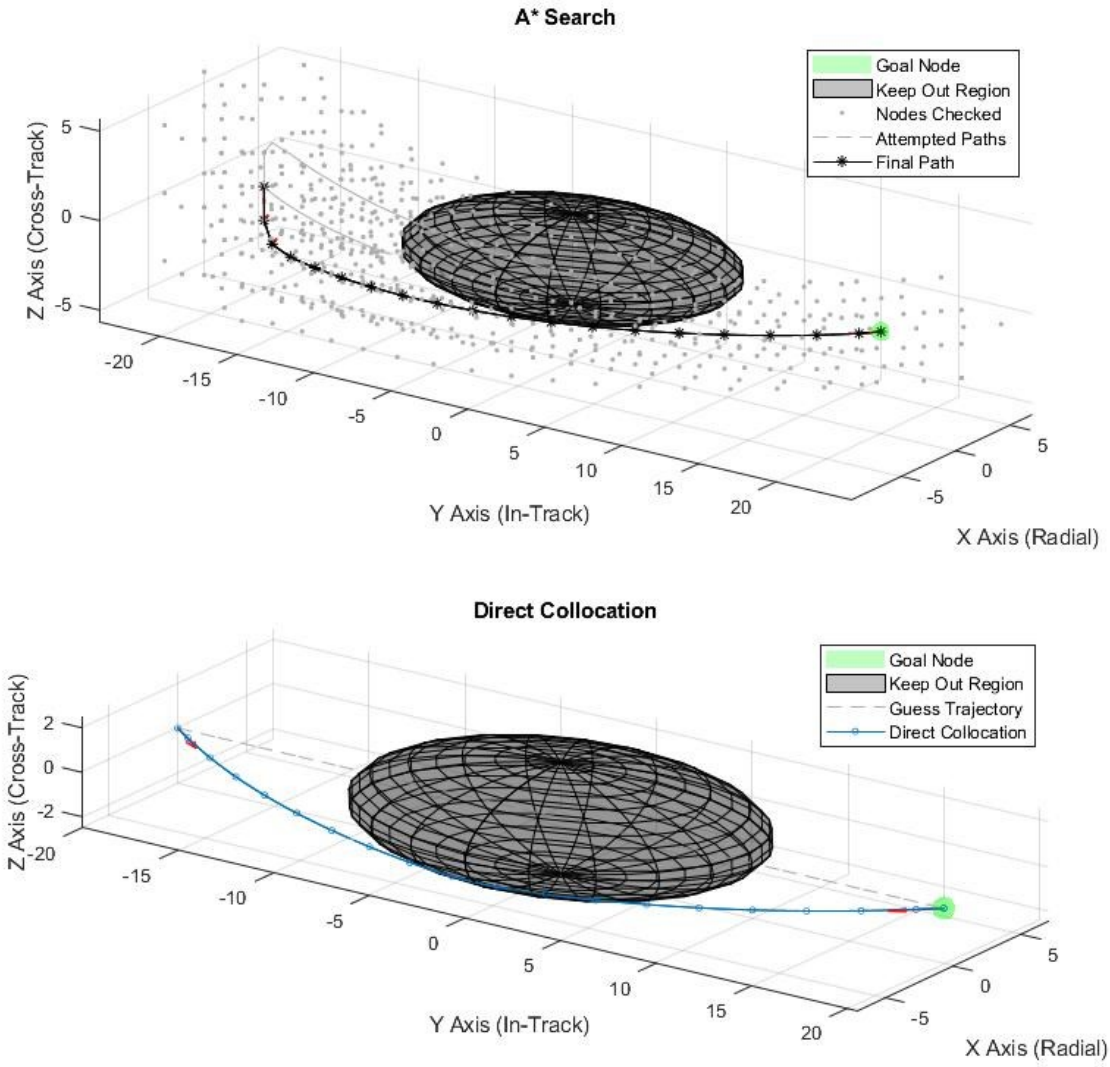

Figure 9: Trajectory solutions to the simple waypoint maneuver example problem. (Left) Trajectory generated via the Adapted A* Search algorithm. (Right) Trajectory generated via the Direct Collocation nonlinear optimization method.

As can be seen above, visually the two trajectories are quite similar. The computational solution time and total delta-V values for each solution method are listed in Table 3 below. As expected, the Direct Collocation nonlinear optimization solution results in a lower delta-V value for the transfer but requires substantially more computational time. The Adapted A* Search algorithm generates a solution that is close to the Direct Collocation solution with a fraction of the

computational time. This example resulted in a higher computational solution time than expected for the Adapted A* Search algorithm. This is likely due to the size of the keep-out-region in the center and its location near the more optimal, low delta-V cost, paths. Different solver parameters for the Adapted A* Search (such as number of nodes, max thrust, etc.) may produce better results.

Table 3: Solution Results for Simple Waypoint Maneuver

Solution Method	Solution time (s)	Total Delta-V (m/s)
Adapted A* Search	10.54	0.2421
Direct Collocation	25.70	0.1577

5.2 Complex Waypoint Maneuver

The second example problem deals with a more complex waypoint maneuver from an initial location of $(-20, -20, -5)$ to a final location of $(0,0,0)$ with three ellipsoid keep out regions and thruster plume impingement constraints. For this example, the thruster plume was modelled as a line segment extending from the location of the burn in the direction of the plume, scaled by the magnitude of the thrust to a maximum of 10 meters (corresponding to maximum thrust at a node). For the Direct Collocation solution method, the plume impingement is checked only at nodal points though continuous thrusting is employed. Additional problem properties are listed in Table 4.

Table 4: Problem Properties for Complex Waypoint Maneuver

Parameter	Chief Orbit (km)	Transfer Time (s)	Follower Mass (kg)	Max Thrust (impulsive) (m/s)	Max Thrust (continuous) (m/s^2)
Value	6791	2400	1 kg	.03	.0002375

Trajectories generated using the two solution methods described above can be seen in Figure 10. The Adapted A* Search algorithm solution is plotted on the left while the Direct Collocation nonlinear optimization method solution is shown on the right.

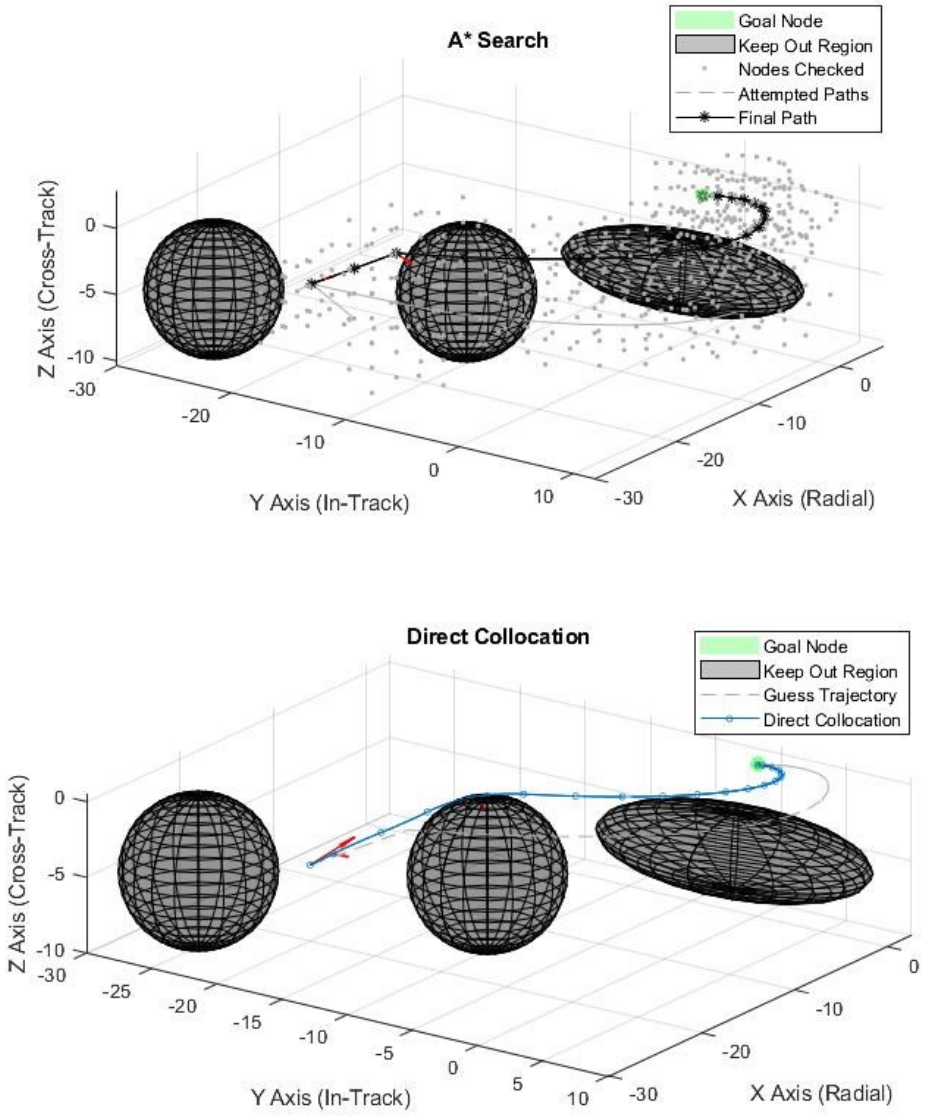

Figure 10: Trajectory solutions to the waypoint maneuver with complex keep out regions and thruster plume impingement constraints. (Left) Trajectory generated via the Adapted A* Search algorithm. (Right) Trajectory generated via the Direct Collocation nonlinear optimization method.

Here the two solution methods produce different trajectories in the path that each takes around the center keep out region though similar delta-V values. The computational solution time for the Adapted A* Search algorithm is much lower than that for the Direct Collocation algorithm. The A* Search algorithm is able to provide a solution in just under 7 seconds which is much faster than

the 32 seconds required by the Direct Collocation solution method. Additionally, the delta-V for the A* Search solution is quite close to that obtained via the Direct Collocation algorithm as can be seen in Table 5. In summary, the Adapted A* Search algorithm performs quite well with complex constraints and provides a very comparable solution in terms of delta-V while maintaining a much shorter computational solution time.

Table 5: Solution Results for Complex Waypoint Maneuver

Solution Method	Solution time (s)	Total Delta-V (m/s)
Adapted A* Search	6.91	0.0708
Direct Collocation	32.04	0.0627

5.3 Injection into NMT Ellipse

This example problem deals with an injection into a natural motion trajectory. The follower attempts to enter into a bounded trajectory about the target, useful for long term station keeping and inspection. This bounded trajectory takes the form of a 2-1 ellipse in the in-track and radial directions respectively and requires no thrusting to maintains. A more extensive discussion on bounded relative motion trajectories is present in section 2.1.4. Additional properties for the problem set-up are listed in Table 6.

Table 6: Problem Properties for Injection into NMT

Parameter	Chief Orbit (km)	Transfer Time (s)	Follower Mass (kg)	Max Thrust (impulsive) (m/s)	Max Thrust (continuous) (m/s^2)
Value	6791	2400	1 kg	.03	.0002375

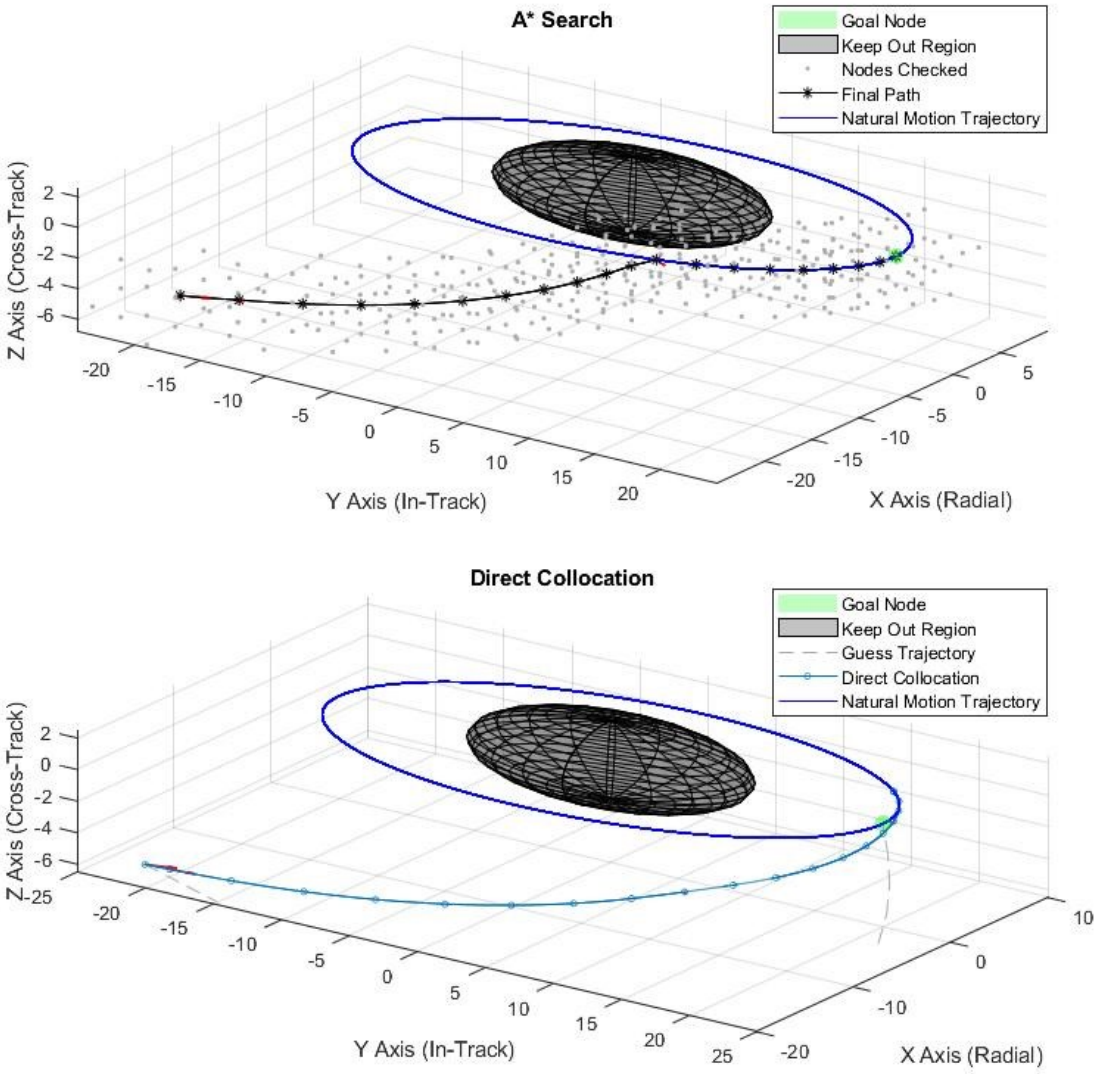

Figure 11: Trajectory solutions to natural motion trajectory example. (Left) Trajectory generated via the Adapted A* Search algorithm. (Right) Trajectory generated via the Direct Collocation nonlinear optimization method.

Again, the two trajectories are quite similar. For this case, the Adapted A* Search method outperforms the nonlinear optimization-based Direct Collocation method in both total delta-V and solution time. This could be due to the nonlinear optimizer getting stuck in a local minimum near its original guess trajectory or from the slight disparity between impulsive and continuous thrusting maximum limits. Exact values for this solution case can be found in Table 7.

Table 7: Solution Results for Natural Motion Trajectory Injection

Solution Method	Solution time (s)	Total Delta-V (m/s)
Adapted A* Search	5.57	0.0486
Direct Collocation	24.57	0.0499

5.4 Algorithm Comparison in 100 Randomized Environments

To provide a better understanding of the performance of each algorithm and variation discussed in this paper 100 randomized environments were generated, and each algorithm was applied to generate a solution for a waypoint maneuver. This set of trials provides insight into the relative performance of the two methods as well as the variation in solution time and delta-V required of the solutions. Four different algorithms were considered, two variations each of the A* Search and Direct Collocation solution methods as described in the following section.

5.4.1 Solution Methods

Two variations of the A* Search algorithm are implemented in this section for better understanding of the benefits of the delta-V heuristic based on the convex optimization sub problem vs the CW Targeting heuristic, as described in section 3.2.2. Two variations of the Direct Collocation solution method are also implemented. The solution methods are described here in the order that they appear in the results: the first solution method is Direct Collocation with the initial guess provided by the CW Targeting algorithm (as discussed in section 2.1.3) denoted as "CW DC." The second method is the A* Search method utilizing the CW Targeting heuristic as described in section 3.2.2 ("CW A*"). The third method is the Adapted A* Search utilizing the convex optimization sub-problem as a heuristic ("A*"). Lastly, the fourth method is the Direct Collocation solution method using the result from the Adapted A* Search as the initial guess trajectory ("A* DC").

5.4.2 Problem to be Solved

For each trial, the same waypoint maneuver as in section 5.2 was attempted by each of the solvers in succession for each of the 100 randomized environments. The environments included 5 spherical keep-out-regions with a 5m radius that were generated and placed at random within the solution space. Thrust plume impingement constraints as described in section 5.2 were also implemented for each of the trial runs. An example of one such randomized environment with obstacles is shown in the following figure.

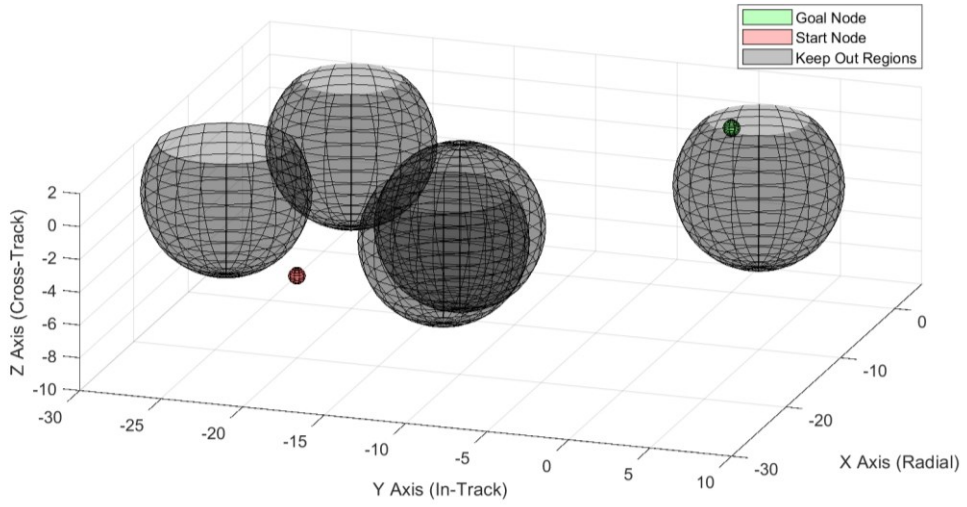

Figure 12: Problem set-up for a complex waypoint maneuver in an example randomized environment. The goal node lies at $(0, 0, 0)$ and the start node at $(-20, -20, -5)$ in the relative reference frame.

Additional information about the problem setup can be found in Table 8.

Table 8: Problem Properties for 100 Runs

Parameter	Chief Orbit (km)	Transfer Time (s)	Follower Mass (kg)	Max Thrust (impulsive) (m/s)	Max Thrust (continuous) (m/s^2)
Value	6791	2400	1 kg	.03	.0002375

5.4.3 Results

Aggregate results for the randomized environment trial runs are presented below. A separate box and whisker plot is generated for computational time and delta-V with each of the solution methods considered. The red line within the box represents the median, and the top and bottom edges of the blue box indicate the 75th and 25th percentiles respectively. Whiskers extend to the most extreme data points not considered outliers and outliers are plotted using a red plus symbol. Whisker length (and the cutoff for a data point being considered an outlier) is set to 1.5 times the interquartile range (middle 50th percentile within the box). This value for whisker length corresponds to ~2.7σ and 99.3% coverage if the data is distributed normally [78].

Computational Time

Results for computational time are shown below in Figure 13.

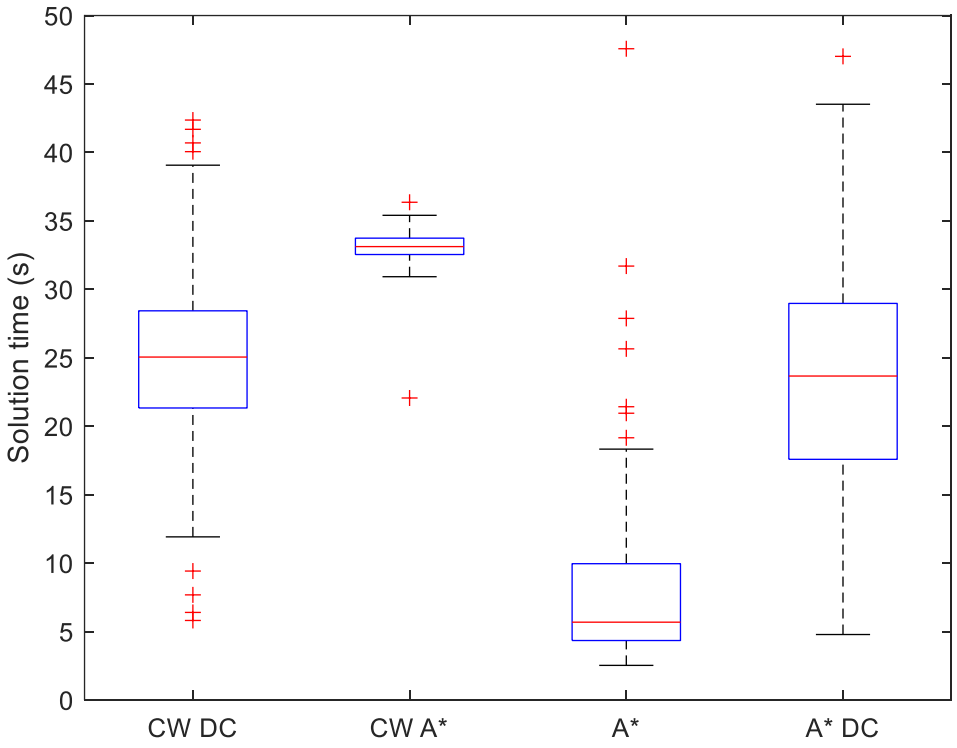

Figure 13: Comparison of solution times required of the various methods for 100 random environments using a box and whisker plot.

Figure 13 shows a clear advantage in terms of computational solution time for the Adapted A* Search algorithm utilizing the convex optimization sub problem as the guiding heuristic, denoted A* in the figure. Between the two A* implementations, perhaps counter intuitively, the A* implementation with the more expensive heuristic estimate produced, on average, solutions with a much lower computational time. This is mainly due to the difference in the number of nodes checked between the CW A* and A* algorithms. Because the A* algorithm utilizes an admissible heuristic as described in section 3.2.2 far fewer nodes need be evaluated. Additionally, the convex optimization sub-problem heuristic also returns the optimal delta-V burn at each node which is added to the number of burns propagated for dynamic grid generation. Adding the optimal burn helps convergence and is not possible in the CW A* method due to restrictions on the delta-V burn magnitudes at each node.

Comparison of the two Direct Collocation methods, each utilizing a different method for generating the initial guess trajectory, provides some insights. The Direct Collocation algorithm using the CW Targeter as the initial guess, CW DC, recorded slightly slower solution times (not including the computational time of the guess algorithm) than that using the A* as the initial guess. Counterintuitively the CW DC method produced more consistent solution times than the A* DC method. This may be attributed to the potential for a larger number of discontinuities associated with multiple impulsive burns in the A* trajectory (as opposed to one large discontinuity at the beginning and end for the CW Targeter guess). It may also be that the nonlinearity of the local solution environment is more important for quicker convergence than proximity to the globally optimal solution, or even starting with a valid (no constraints violated) trajectory. It is possible that the CW Targeter provides an initial guess that is more likely to be in a more convex local minimum, and the solver is able to converge in a more consistent solution time.

Comparing the Direct Collocation algorithms and the A* Algorithms a few important observations emerge. Firstly, solution times for the Adapted A* Search utilizing the convex optimization heuristic were much lower, on average, than those from either of the Direct Collocation methods. Secondly, the Adapted A* Search algorithm produced a smaller range of solution times associated with more consistent solution times (e.g., less variability). This reduced variability is important

when considering an algorithm for autonomous on-board application. In Addition, 7 outliers were identified among the solution times for the Adapted A* Search algorithm. This may be unfavorable for on-board applications, but it should be noted that 6 of the 7 outliers still lie within the range of the Direct Collocation solution times. The potentially most important result is not apparent in the box and whisker plot. 2 of the 100 trial runs with the CW DC method using the nonlinear solver did not converge to a solution. This may be a rare occurrence but represents a total failure of a path planning algorithm – to provide no solution and no information on whether a solution exists. For comparison, the CW A*, A*, and A* DC methods were able to produce a solution for every trial. This highlights one additional benefit of the combined A* DC solution methodology – starting with a viable solution likely improves the chances that the Direct Collocation nonlinear solver is able to converge, and in the event that the Direct Collocation solver is unable to converge, an acceptable trajectory solution already exists (from the A* method).

Delta-V

A comparison of the four methods in terms of delta-V requirements is shown in the box and whisker plot below.

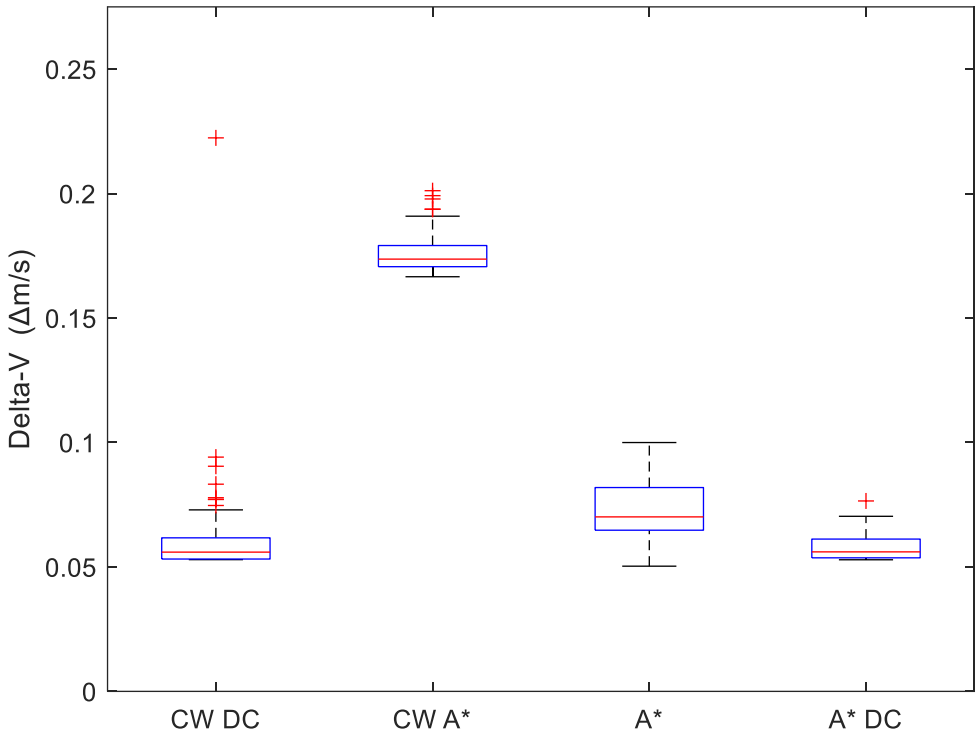

Figure 14: Box and whisker plot for the delta-V values of the various methods for 100 random environments

The results for delta-V shown in Figure 14 are more in line with expectations for the performance of each algorithm. The nonlinear optimization based Direct Collocation methods appear to perform the better than the other in terms of delta-V. Between the two A* Search methods used, the A* using the convex optimization heuristic produced a much lower median delta-V value and a fairly consistent spread. The CW A* method produced comparatively large delta-V values (nearly twice that of the Adapted A* Search), and a very small interquartile range. These results highlight the benefits of calculating and being able to include the optimal delta-V burn at each node as is done in the A* Search method using the convex optimization sub problem. This results in a marked improvement in terms of delta-V between the two A* Search based methods.

There is little noticeable difference between the two Direct Collocation methods used. The A* DC method produced nearly identical median delta-V values and interquartile ranges to the CW DC method. This result is somewhat unexpected as the CW Targeter was thought to generally provide a worse guess at the initial trajectory. The CW Targeter should produce a solution less likely to be near the optimal solution (and allow the nonlinear solver to converge to lower delta-V solutions) than the A* method which provides a globally optimal minimum delta-V trajectory to the level of discretization. It could be, however, that the CW Targeter generally provides an initial guess that is "good enough" for the Direct Collocation method to converge to a low cost solution most of the time. This likely also depends on the initial positions and time of flight chosen for the waypoint maneuver – in some cases the CW Targeter may produce a much worse initial guess. Additionally, for 2 of the 100 solution runs the CW DC method failed to converge to any solution while the A* DC method was able to converge. This reveals some benefit to using the Adapted A* Search as a guess method over the CW Targeter even though on average delta-V values were similar.

Delta-V vs Computational Time

Two plots of delta-V vs computational time for the A* and Direct Collocation algorithms is shown below for better understanding of the performance of each individual algorithm.

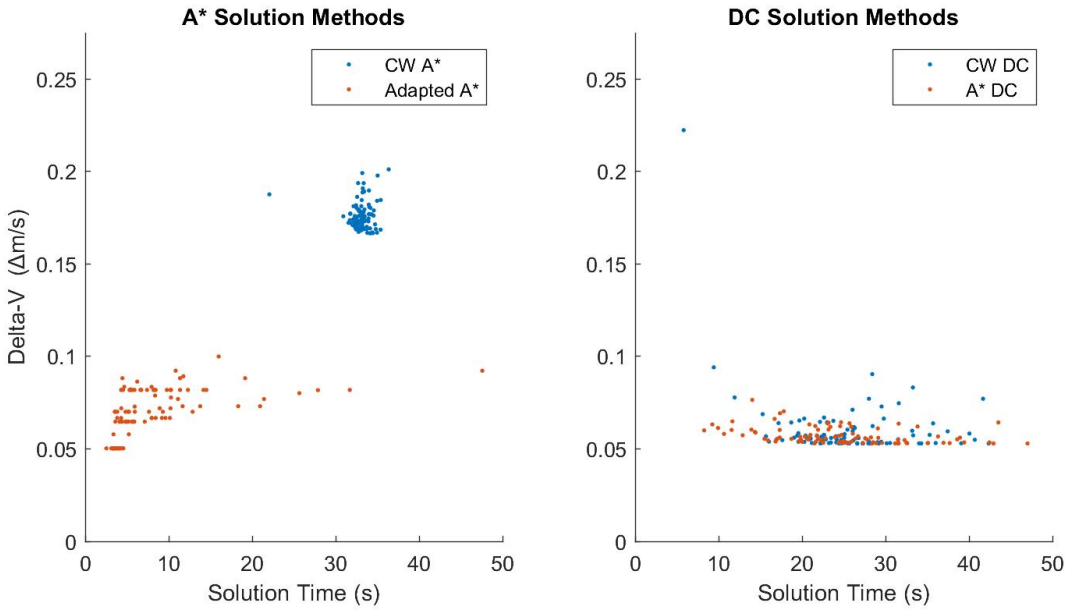

Figure 15: Comparison plot of the delta-V vs computational solution time for the A* Search and Direct Collocation algorithms in each of the 100 random environments

The above plot represents expected results. The CW A* Search algorithm has a tight grouping of both solution time and delta-V. The results looking at the Adapted A* Search algorithm are a little more interesting. Firstly, it appears that quicker solution times are correlated with lower delta-V solutions. This makes sense if you consider the best case (or near best case) scenario in which the optimal solution path as solved with the convex optimization sub problem is unobstructed by constraints. In this case, the least number of nodes is required to reach the goal which results in the lowest possible computational time required to produce a solution. More nodes are searched each time the solver is required to diverge from this optimal path, leading to both higher delta-V values and solution times. Secondly, the delta-V results from the Adapted A* Search appear to be somewhat quantized in that some distinct delta-V levels can be observed in Figure 15. This may be a result of the fixed magnitudes of the "sub-optimal" delta-V burns considered at each node for grid generation. As one of these "sub-optimal" (with respect to the unconstrained case) burns is taken, the delta-V required to reach the goal is increased by some finite value. The solver (due to the nature of the problem set-up) may be more inclined to take a set number paths that represent the more optimal paths (in terms of combinations of random burns and optimal path arcs). These close-optimal paths are then the delta-V levels that can be seen in the figure.

Results from the Direct Collocation algorithms are perhaps less interesting. We see a fairly large spread of solution times and fairly consistent delta-V values across the board. In the CW DC case, (the Direct Collocation algorithm using the CW Targeter as the initial guess) there appears to be a larger number of outliers, though these occur throughout the spectrum of solution times.

Chapter 6

Conclusion

6.1 Summary

Both the Adapted A* Search algorithm and the nonlinear optimization based Direct Collocation method for trajectory generation are applicable to the satellite relative motion problem for in-space-inspection. Both solution methods are capable of incorporating multiple complex nonlinear constraints such as multiple keep out regions and thruster plume impingement. Generally, the Adapted A* Search algorithm had significantly lower computation times and the nonlinear optimization Direct Collocation method produced trajectories with slightly lower delta-V values (though not necessarily in every example).

The Adapted A* Search method is quite fast depending on the number of nodal points used and the number of delta-V burns to propagate for child nodes. The Adapted A* Search algorithm also has some nice convergence properties including *resolution completeness* (if a solution exists in our discretization, it will be found in finite time) and *resolution optimality*, that is if a solution is found then that solution is known to be the global optimal to the level of resolution used. This Adapted A* Search method utilizing a novel heuristic function based on the solution to an N-burn convex optimization sub-problem shows drastic improvement over the A* Search method utilizing a CW Targeter as the heuristic. The Adapted A* Search algorithm performs significantly better both in terms of computational time required to generate a solution and in the quality of the solution generated (low delta-V). The efficient utilization of a delta-V based heuristic also likely means that the Adapted A* Search method provides faster and better solutions than other randomness-based path planning algorithms for the spacecraft relative motion problem.

The nonlinear optimization based Direct Collocation method is capable of incorporating almost any constraint (including but not limited to the keep out regions and thruster plume impingement) and capable of numerically finding a solution via a nonlinear programming solver. The nonlinear programming solver, however, is not able to provide any guarantees on finding a solution (i.e., returning a solution if one exists) or ensuring that a solution found is optimal.

For generating relative motion trajectories, either method or a combination of the two can be applied depending on the specific goals of the mission. A combined solution methodology is developed in which an initial solution generated by the Adapted A* Search algorithm is used to seed an initial trajectory for the Direct Collocation. If time permits, this combined solution method may offer a better overall solution and ensure that a solution exists before attempting the longer Direct Collocation method. Both methods were shown to generate low delta-V trajectories though the Adapted A* method produced significantly lower computational times that may better allow for autonomous on-board application.

6.2 Future Work

This work provides the formulation of an Adapted A* Search algorithm and an initial assessment of the algorithm as applied to the satellite relative motion path planning problem. There are, however, several areas in which improvements could be made or additional work done in order to extend the work in this thesis.

Variants of the A* Search method could be applied to extend the applicability of this work. Of note, there is a dynamic replanning version of the A* Search method known as D* that provides more efficient computation in an unknown environment. Work in this thesis may also be applied to a multi-agent version of A* to allow for efficient, low delta-V, trajectory planning for multiple follower satellites.

Additional relative motion problem types could be investigated including additional constraints particularly relating to safety and collision avoidance in the event of a thruster failure. Being able to guarantee collision avoidance and safety under difference circumstances is important to the relative motion trajectory planning problem as a whole. Ensuring long term safety under thruster failure or other uncertainties is essential in developing an autonomous path planning algorithm for on-board applications.

Another area for future work would be an investigation into the average computational solution time and delta-V values across a wider range of problem types between the Adapted A* Search method presented in this work and the nonlinear optimization based Direct Collocation method or additional solution methods. Data for a larger number of trajectory planning problems comparing the Adapted A* Search method and other solution methods would be helpful in drawing extended conclusions on the benefits of each method.

Bibliography

[1] W. Fehse, *Automated Rendezvous and Docking of Spacecraft*, 1st ed. New York: Cambridge University Press, 2003.

[2] M. E. Polites, "An Assessment of the Technology of Automated Rendezvous and Capture in Space," Mar. 1998.

[3] M. E. Polites, "Technology of Automated Rendezvous and Capture in Space," *J. Spacecr. Rockets*, vol. 36, no. 2, pp. 280–291, 1999.

[4] D. A. Whelan, E. A. Adler, S. B. Wilson III, and G. M. Roesler, Jr., "DARPA Orbital Express program: effecting a revolution in space-based systems," in *Small Payloads in Space*, 2000, vol. 4136, no. November 2000, pp. 48–56.

[5] "2020 NASA Technology Taxonomy," 2020.

[6] D. C. Woffinden and D. K. Geller, "Navigating the road to autonomous orbital rendezvous," *J. Spacecr. Rockets*, vol. 44, no. 4, pp. 898–909, 2007.

[7] R. Z. Sagdeev and A. V Zakharov, "Brief history of the Phobos mission," *Nature*, vol. 341, no. 6243, pp. 581–585, 1989.

[8] T. Kubota, T. Hashimoto, J. N. I. Kawaguchi, M. Uo, and K. N. I. Shirakawa, "Guidance and Navigation of Hayabusa Spacecraft for Asteroid Exploration and Sample Return Mission," in *2006 SICE-ICASE International Joint Conference*, 2006, pp. 2793–2796.

[9] A. B. Bosse *et al.*, "SUMO: Spacecraft for the Universal Modification of Orbits," in *Spacecraft Platforms and Infrastructure*, 2004, vol. 5419, no. August 2004, pp. 36–46.

[10] R. W. Madison, "Micro-satellite Based On-Orbit Servicing Work at the Air Force Research Laboratory," in *2000 IEEE Aerospace Conference. Proceedings (Cat. No.00TH8484)*, 2000, vol. 4, pp. 215–226 vol.4.

[11] B. Dunbar, “DART,” 2007. [Online]. Available: https://www.nasa.gov/mission_pages/dart/main/.

[12] E. Gill, S. D’Amico, and O. Montenbruck, “Autonomous Formation Flying for the PRISMA Mission,” *J. Spacecr. Rockets*, vol. 44, no. 3, pp. 671–681, 2007.

[13] P. Bodin, R. Larsson, F. Nilsson, C. Chasset, R. Noteborn, and M. Nylund, “PRISMA : An In-Orbit Test Bed for Guidance , Navigation , and Control Experiments,” vol. 46, no. 3, 2009.

[14] M. Younis *et al.*, “TanDEM-X: A Satellite Formation for High-Resolution Radar Interferometry,” *Proc. Int. Radar Symp.*, vol. 2007-Janua, no. 11, pp. 3317–3341, 2007.

[15] D. (DARPA) Barnhart *et al.*, “Phoenix Program Status - 2013,” *AIAA Sp. 2013 Conf. Expo.*, no. September, 2013.

[16] A. (DARPA) Saplan, “Robotic Servicing of Geosynchronous Satellites (RSGS).” [Online]. Available: https://www.darpa.mil/program/robotic-servicing-of-geosynchronous-satellites.

[17] “Northrop Grumman and Intelsat Make History with Docking of Second Mission Extension Vehicle to Extend Life of Satellite,” 2021. [Online]. Available: https://www.intelsat.com/newsroom/northrop-grumman-and-intelsat-make-history-with-docking-of-second-mission-extension-vehicle-to-extend-life-of-satellite/.

[18] “SpaceLogistics,” *2021*. [Online]. Available: https://www.northropgrumman.com/space/space-logistics-services/.

[19] “Spheres,” 2018. [Online]. Available: https://www.nasa.gov/spheres/home.

[20] E. Stoll, S. Jaekel, J. Katz, A. Saenz-Otero, and R. Varatharajoo, “SPHERES Interact-Human-Machine Interaction aboard the International Space Station,” *J. F. Robot.*, vol. 29, no. 4, pp. 554–575, Jul. 2012.

[21] "Aerocubes," 2018. [Online]. Available: https://aerospace.org/sites/default/files/2018-06/Aerocubes0318.pdf.

[22] J. Crusan, "Future Human Exploration Planning: Lunar Orbital Platform-Gateway and Science Workshop Findings," 2018.

[23] J. McGuire, "OSAM-1: On-Orbit Servicing, Assembly and Manufacturing - 1." [Online]. Available: https://nexis.gsfc.nasa.gov/osam-1.html.

[24] R. Reesman and A. Rogers, "Getting in Your Space: Learning from Past Rendezvous and Proximity Operations," 2018.

[25] J. Davis, J. Mayberry, and J. Penn, "On-orbit Servicing: Inspection, Repair, Refuel, Upgrade, and Assembly of Satellites in Space," no. April, pp. 1–14, 2019.

[26] J. S. Hudson and D. Kolosa, "Versatile On-Orbit Servicing Mission Design in Geosynchronous Earth Orbit," *J. Spacecr. Rockets*, vol. 57, no. 4, pp. 844–850, 2020.

[27] W. Fehse, "Close Range Rendezvous Operations," in *Automated Rendezvous and Docking of Spacecraft*, 1st ed., M. J. Rycroft and W. Shyy, Eds. New York: Cambridge University Press, 2003, pp. 19–24.

[28] G. Dettleff, "Plume Flow and Plume Impingement in Space Technology," *Prog. Aerosp. Sci.*, vol. 28, no. 1, pp. 1–71, 1991.

[29] D. A. Vallado and W. D. McClain, *Fundamentals of Astrodynamics and Applications*. Springer Netherlands, 2001.

[30] K. T. Alfriend, S. R. Vadali, P. Gurfil, J. P. How, and L. S. Breger, "Chapter 5 - Linear Equations of Relative Motion," in *Spacecraft Formation Flying*, K. T. Alfriend, S. R. Vadali, P. Gurfil, J. P. How, and L. S. Breger, Eds. Oxford: Butterworth-Heinemann, 2010, pp. 83–121.

[31] F. De Bruijn, E. Gill, and J. How, "Comparative Analysis of Cartesian and Curvilinear Clohessy-Wiltshire Equations," *J. Aerosp. Eng. Sci. Appl.*, vol. 3, no. 2, 2011.

[32] J. Sullivan, S. Grimberg, and S. D'Amico, "Comprehensive Survey and Assessment of Spacecraft Relative Motion Dynamics Models," *J. Guid. Control. Dyn.*, vol. 40, no. 8, pp. 1837–1859, 2017.

[33] W. H. Clohessy and R. S. Wiltshire, "Terminal Guidance System for Satellite Rendezvous," *J. Aerosp. Sci.*, vol. 27, no. 9, pp. 653–658, 1960.

[34] W. E. Wiesel, *Spaceflight Dynamics*, 3rd ed. CreateSpace, 2010.

[35] T. A. Lovell and D. A. Spencer, "Relative Orbital Elements Formulation based upon the Clohessy-Wiltshire Equations," *J. Astronaut. Sci.*, vol. 61, no. 4, pp. 341–366, 2014.

[36] A. V. Rao, "A Survey of Numerical Methods for Optimal Control," *Adv. Astronaut. Sci.*, vol. 135, 2010.

[37] B. A. Conway, "A Survey of Methods Available for the Numerical Optimization of Continuous Dynamic Systems," *J. Optim. Theory Appl.*, vol. 152, no. 2, pp. 271–306, 2012.

[38] J. T. Betts, "Survey of Numerical Methods for Trajectory Optimization," *J. Guid. Control. Dyn.*, vol. 21, no. 2, pp. 193–207, 1998.

[39] M. Kelly, "An Introduction To Trajectory Optimization: How To Do Your Own Direct Collocation," *SIAM Rev.*, vol. 59, no. 4, pp. 849–904, 2017.

[40] G. Q. Huang, Y. P. Lu, and Y. Nan, "A Survey of Numerical Algorithms for Trajectory Optimization of Flight Vehicles," *Sci. China Technol. Sci.*, vol. 55, no. 9, pp. 2538–2560, 2012.

[41] F. Topputo and C. Zhang, "Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications," *Abstr. Appl. Anal.*, vol. 2014, 2014.

[42] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path Planning for Autonomous Driving in Unknown Environments," in *Experimental Robotics*, 2009, pp. 55–64.

[43] C. Goerzen, Z. Kong, and B. Mettler, “A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 57, no. 1–4, pp. 65–100, 2010.

[44] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, “Survey of Robot 3D Path Planning Algorithms,” *J. Control Sci. Eng.*, vol. 2016, 2016.

[45] Y. Yang, J. Pan, and W. Wan, “Survey of Optimal Motion Planning,” *IET Cyber-Systems Robot.*, vol. 1, no. 1, pp. 13–19, 2019.

[46] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, “Spacecraft Trajectory Planning with Avoidance Constraints using Mixed-Integer Linear Programming,” *J. Guid. Control. Dyn.*, vol. 25, no. 4, pp. 755–764, 2002.

[47] A. Richards and J. P. How, “Model Predictive Control of Vehicle Maneuvers with Guaranteed Completion Time and Robust Feasibility,” *Proc. Am. Control Conf.*, vol. 5, pp. 4034–4040, 2003.

[48] N. G. Ortolano, “Autonomous Trajectory Planning for Satellite RPO and Safety of Flight Using Convex Optimization,” Utah State University, 2018.

[49] N. Ortolano, D. K. Geller, and A. Avery, “Autonomous Optimal Trajectory Planning for Orbital Rendezvous, Satellite Inspection, and Final Approach Based on Convex Optimization,” *J. Astronaut. Sci.*, pp. 444–479, 2021.

[50] B. A. Conway, “The Problem of Spacecraft Trajectory Optimization,” in *Spacecraft Trajectory Optimization*, B. A. E. Conway, Ed. Cambridge University Press, 2010, pp. 1–15.

[51] E. R. Prince, “Optimal Finite Thrust Guidance Methods for Constrained Satellite Proximity Operations Inspection Maneuvers,” Air Force Institute of Technology, 2018.

[52] M. Pontani and B. A. Conway, “Particle Swarm Optimization applied to Space Trajectories,” *J. Guid. Control. Dyn.*, vol. 33, no. 5, pp. 1429–1441, 2010.

[53] M. Pontani and B. A. Conway, “Optimal Finite-Thrust Rendezvous Trajectories found via Particle Swarm Algorithm,” *J. Spacecr. Rockets*, vol. 50, no. 6, pp. 1222–1234, 2013.

[54] K. M. Nastasi, D. J. Thomas, K. Tetreault, I. Elliott, and J. T. Black, “Real-Time Optimal Control & Tracking of Autonomous Micro-Satellite Proximity Operations,” *AIAA Sp. Astronaut. Forum Expo. Sp. 2016*, no. September, 2016.

[55] D. J. Showalter, “Optimal Autonomous Spacecraft Resiliency Maneuvers using Metaheuristics,” p. 486, 2014.

[56] D. J. Showalter and J. T. Black, “Optimal Continuous-Thrust Responsive Theater Maneuvers,” *J. Spacecr. Rockets*, vol. 52, no. 5, pp. 1375–1387, 2015.

[57] L. J. Digirolamo, “A Hybrid Stochastic Motion Planning Algorithm for Safe and Efficient, Close Proximity, Autonomous Spacecraft Missions,” Pennsylvania State University, 2014.

[58] J. A. Starek, “Sampling-Based Motion Planning for Safe and Efficient Spacecraft Proximity Operations,” Stanford University, 2016.

[59] C. K. Niiya, “An Application of the A* Search to Trajectory Optimization,” Massachusetts Institute of Technology, 1990.

[60] J. F. Raquet, “Six Degree of Freedom Trajectory Planner for Spacecraft Proximity Operations Using an A* Node Search,” Massachusetts Institute of Technology, 1991.

[61] M. C. Jackson, “A Six Degree of Freedom Plume-Fuel Optimal Trajectory Planner for Spacecraft Proximity Operations Using an A* Node Search,” Massachusetts Institute of Technology, 1994.

[62] J. M. Phillips, L. E. Kavraki, and N. Bedrossian, “Probabilistic Optimization Applied to Spacecraft Rendezvous and Docking,” in *Advances in the Astronautical Sciences*, 2003, vol. 114 I, pp. 261–274.

[63] G. Saive and M. Vasile, “Probabilistic Optimisation applied to Spacecraft Rendezvous on Keplerian Orbits,” 2004.

[64] M. A. Paluszek and S. J. Thomas, “Generalized 3D spacecraft Proximity Path Planning using A*,” in *Collection of Technical Papers - InfoTech at Aerospace: Advancing Contemporary Aerospace Technologies and Their Integration*, 2005, vol. 2, no. September, pp. 1230–1239.

[65] G. D. Francis, E. G. Collins, O. Chuy, and A. Sharma, “Sampling-Based Trajectory Generation for Autonomous Spacecraft Rendezvous and Docking,” in *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013.

[66] F. Baldini *et al.*, “Fast Motion Planning for Agile Space Systems with Multiple Obstacles,” 2016, no. September, pp. 1–14.

[67] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, “An Admissible Heuristic to Improve Convergence in Kinodynamic Planners Using Motion Primitives,” *IEEE Control Syst. Lett.*, vol. 4, no. 1, pp. 175–180, 2020.

[68] A. C. Rogers, “Optimization-Based Guidance For Satellite Relative Motion,” Virginia Polytechnic Institute and State University, 2016.

[69] L. W. Neustadt, “Optimization, A Moment Problem, and Nonlinear Programming,” *SIAM Control*, vol. 2, no. 1, pp. 33–53, 1964.

[70] S. P. Hughes, L. M. Mailhe, and J. J. Guzman, “A Comparison of Trajectory Optimization Methods for the Impulsive Minimum Fuel Rendezvous Problem,” in *26th Annual Guidance and Control Conference*, 2003.

[71] J. E. Prussing, “Optimal Four-Impulse Fixed-Time Rendezvous in the Vicinity of a Circular Orbit,” *AIAA J.*, vol. 7, no. 5, pp. 928–935, 1969.

[72] J. E. Prussing, “Optimal Two- and Three-Impulse Fixed-Time Rendezvous in the Vicinity of a Circular Orbit,” *AIAA J.*, vol. 8, no. 7, pp. 1221–1228, 1970.

[73] J. E. Prussing and J. H. Chiu, “Optimal Multiple-Impulse Time-Fixed Rendezvous Between Circular Orbits,” *J. Guid. Control. Dyn.*, vol. 9, no. 1, pp. 17–22, 1986.

[74] T. E. Carter, “Fuel-Optimal Maneuvers of a Spacecraft Relative to a Point in Circular Orbit,” *J. Guid. Control. Dyn.*, vol. 7, no. 6, pp. 710–716, 1984.

[75] T. E. Carter, “Quadratic-Based Computation of Four-Impulse Optimal Rendezvous near Circular Orbit,” vol. 23, no. 1, pp. 109–117, 2000.

[76] A. L. Herman and B. A. Conway, “Direct optimization using collocation based on high-order Gauss-Lobatto quadrature rules,” *J. Guid. Control. Dyn.*, vol. 19, no. 3, pp. 592–599, 1996.

[77] “Fmincon,” *MathWorks*, 2021. [Online]. Available: https://www.mathworks.com/help/optim/ug/fmincon.html. [Accessed: 06-Apr-2021].

[78] “boxplot,” *MathWorks*, 2021. [Online]. Available: https://www.mathworks.com/help/stats/boxplot.html. [Accessed: 22-Aug-2021].